

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SDIR

OBJECT MODULE PLACED IN MAIN80

COMPILER INVOKED BY: PLM80 MAIN80.PL M DEBUG PAGEWIDTH(130) OPTIMIZE OBJECT(MAIN80)

```

1      $title ('SDIR 8080 - Main Module')
      sdirc:                                     /* SDIR FOR 8080 */
      do;

      $include(copyrt.lit)

      =
      = /*
      =   Copyright (C) 1982
      =   Digital Research
      =   P.O. Box 579
      =   Pacific Grove, CA 93950
      =   */
      =

2 1    declare plm label public;

      $include(main.plm)

      =
      = /* C P / M - M P / M   DIRECTORY COMMON (SDIR) */
      =
      = /* BEGINNING OF COMMON MAIN MODULE */
      =
      = /* This module is included in main80.plm or main86.plm. */
      = /* The differences between 8080 and 8086 versions are */
      = /* contained in the modules main80.plm, main86.plm and */
      = /* dpb80.plm, dpb86.plm and the submit files showing */
      = /* the different link and location addresses.          */
      =
      =
      = $include (comlit.lit)

3 1    =1 declare
      =1      lit      literally      'literally',
      =1      dcl      lit      'declare',
      =1      true     lit      'Offh',
      =1      false    lit      '0',
      =1      boolean  lit      'byte',
      =1      forever  lit      'while true',
      =1      cr       lit      '13',
      =1      lf       lit      '10',
      =1      tab      lit      '9',
      =1      ctrlc    lit      '3',
      =1      ff       lit      '12',
      =1      page$len$offset lit      '1ch',
      =1      nopage$mode$offset lit      '2Ch',
      =1      sectorlen lit      '128';
      = $include (mon.plm)
      =1
      =1 /* definitions for assembly interface module */

4 1    =1 declare

```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

      =1     fcb (33) byte external,     /* default file control block     */
      =1     maxb address external,     /* top of memory     */
      =1     buff(128) byte external;     /* default buffer     */
      =1
5   1   =1   mon1: procedure(f,a) external;
6   2   =1       declare f byte, a address;
7   2   =1       end mon1;
      =1
8   1   =1   mon2: procedure(f,a) byte external;
9   2   =1       declare f byte, a address;
10   2   =1       end mon2;
      =1
11   1   =1   mon3: procedure(f,a) address external;
12   2   =1       declare f byte, a address;
13   2   =1       end mon3;
      =1
      =
      =
14   1   =   dcl patch (128) address;
      =
      =     /* Scanner Entry Points in scan.plm */
      =
15   1   =   scan: procedure(pcb$adr) external;
16   2   =       declare pcb$adr address;
17   2   =   end scan;
      =
18   1   =   scan$init: procedure(pcb$adr) external;
19   2   =       declare pcb$adr address;
20   2   =   end scan$init;
      =
      =     /* ----- Routines in other modules ----- */
      =
21   1   =   search$init: procedure external;     /* initialization of search.plm */
22   2   =   end search$init;
      =
23   1   =   get$files: procedure external;     /* entry to search.plm */
24   2   =   end get$files;
      =
25   1   =   sort: procedure external;     /* entry to sort.plm */
26   2   =   end sort;
      =
27   1   =   mult23: procedure (num) address external;     /* in sort.plm */
28   2   =   dcl num address;
29   2   =   end mult23;
      =
30   1   =   display$files: procedure external;     /* entry to disp.plm */
31   2   =   end display$files;
      =
      =     /* ----- Routines in util.plm ----- */
      =
32   1   =   printb: procedure external;
33   2   =   end printb;
      =
34   1   =   print$char: procedure(c) external;
35   2   =   dcl c byte;
36   2   =   end print$char;
      =

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

37 1 = print: procedure(string$adr) external;
38 2 = dcl string$adr address;
39 2 = end print;
    =
40 1 = crlf: procedure external;
41 2 = end crlf;
    =
42 1 = p$decimal: procedure(value,fieldsize,zsup) external;
43 2 =     dcl value address,
    =         fieldsize address,
    =         zsup boolean;
44 2 = end p$decimal;
    =
    = /* ----- */
    =
45 1 = dcl debug boolean public initial (false);
    =
    = /* ----- version information ----- */
    =
46 1 = dcl (os,bdos) byte public;
    = $include (vers.lit)
47 1 =1 declare
    =1     bdos20 lit '20h',
    =1     bdos30 lit '30h',
    =1     apm    lit '01h',
    =1     apm86  lit '11h';
    =
    = $include (fcb.lit)
    =1
48 1 =1 declare
    =1     f$drvusr    lit '0',    /* drive/user byte      */
    =1     f$name     lit '1',    /* file name            */
    =1     f$namelen  lit '8',    /* file name length     */
    =1     f$type     lit '9',    /* file type field      */
    =1     f$typeplen lit '3',    /* type length          */
    =1     f$rw       lit '9',    /* high bit is R/W attribute */
    =1     f$dirsyz  lit '10',   /* high bit is dir/sys attribute */
    =1     f$arcc    lit '11',   /* high bit is archive attribute */
    =1     f$ex      lit '12',   /* extent                */
    =1     f$s1      lit '13',   /* module byte           */
    =1     f$rc      lit '15',   /* record count          */
    =1     f$diskmap lit '16',   /* file disk map         */
    =1     diskmaplen lit '16',  /* disk map length       */
    =1     f$drvusr2  lit '16',  /* fcb2                  */
    =1     f$name2    lit '17',
    =1     f$type2    lit '25',
    =1     f$rrec     lit '33',   /* random record         */
    =1     f$rreco   lit '35',   /* ' ' overflow          */
    =
    = $include (search.lit)
    =1
49 1 =1 declare /* what kind of file user wants to find */
    =1     find$structure lit 'structure (
    =1     dir byte,
    =1     sys byte,

```

CP/M Plus Ver 3.0

COPYRIGHT © 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

CP3-135

```

      =1      ro byte,
      =1      rw byte,
      =1      pass byte,
      =1      xfc byte,
      =1      nonxfc byte,
      =1      exclude byte)';
      =1
50 1  =1 declare
      =1      max$search$files literally '10';
      =1
51 1  =1 declare
      =1      search$structure lit 'structure(
      =1      drv byte,
      =1      name(8) byte,
      =1      type(3) byte,
      =1      anyfile boolean)';      /* match on any drive if true */
      =1
      =
52 1  = dcl find find$structure public initial
      =      (false,false,false,false, false,false,false,false);
      =
53 1  = dcl
      =      num$search$files byte public initial(0),
      =      no$page$mode byte public initial(0),
      =      search (max$search$files) search$structure public;
      =
54 1  = dcl first$$i$adr address external;
55 1  = dcl get$all$dir$entries boolean public;
56 1  = dcl first$pass boolean public;
      =
57 1  = dcl usr$vector address public initial(0),      /* bits for user $s to scan */
      =      active$usr$vector address public,      /* active users on curdrv */
      =      drv$vector address initial (0);      /* bits for drives to scan */
      =
      = $include (format.lit)
      =1
58 1  =1 dcl form$short lit '0',      /* format values for SDIR */
      =1      form$size lit '1',
      =1      form$full lit '2';
      =1
      =
59 1  = dcl format byte public initial (form$full),
      =      page$len address public initial (0ffffh),
      =      /* lines on a page before printing new headers, 0 forces initial hdrs */
      =      message boolean public initial(false), /* show titles when no files found */
      =      formfeeds boolean public initial(false), /* use form feeds */
      =      date$opt boolean public initial(false), /* dates display */
      =      display$attributes boolean public initial(false); /* attributes display */
      =
60 1  = dcl file$displayed boolean external;
      =      /* true if 1 or more files displayed by dsh.plm */
      =
61 1  = dcl sort$op boolean initial (true);      /* default is to do sorting */
62 1  = dcl sorted boolean external;      /* if successful sort */
      =
      =
63 1  = dcl cur$usr byte public,      /* current user being searched */

```

COPYRIGHT © 1932

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

      =      cur$drv byte public;      /* current drive      */
      =
      =      /* ----- BDOS calls ----- */
      =
64  1 =      get$version: procedure address; /* returns current version information */
65  2 =          return mon2(12,0);
66  2 =      end get$version;
      =
67  1 =      select$drive: procedure(d);
68  2 =          declare d byte;
69  2 =          call mon1(14,d);
70  2 =      end select$drive;
      =
71  1 =      search$first: procedure(d) byte external;
72  2 =          dcl d address;
73  2 =      end search$first;
      =
74  1 =      search$next: procedure byte external;
75  2 =      end search$next;
      =
76  1 =      get$cur$drv: procedure byte;      /* return current drive number */
77  2 =          return mon2(25,0);
78  2 =      end get$cur$drv;
      =
79  1 =      getlogin: procedure address;      /* get the login vector */
80  2 =          return mon3(24,0);
81  2 =      end getlogin;
      =
82  1 =      getusr: procedure byte;      /* return current user number */
83  2 =          return mon2(32,0ffh);
84  2 =      end getusr;
      =
85  1 =      getscbbyte: procedure (offset) byte;
86  2 =          declare offset byte;
87  2 =          declare scbpb structure
      =              (offset byte,
      =                  set    byte,
      =                  value address);
88  2 =          scbpb.offset = offset;
89  2 =          scbpb.set = 0;
90  2 =          return mon2(49,,scbpb);
91  2 =      end getscbbyte;
      =
92  1 =      set$console$mode: procedure;
      =          /* set console mode to control-c only */
93  2 =          call mon1(109,1);
94  2 =      end set$console$mode;
      =
95  1 =      terminate: procedure public;
96  2 =          call mon1 (0,0);
97  2 =      end terminate;
      =
      =      /* ----- Utility routines ----- */
      =
98  1 =      number: procedure (char) boolean;
99  2 =          dcl char byte;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

100 2 =      return(char >= '0' and char <= '9');
101 2 =      end number;
    =
102 1 =      make$numeric: procedure(char$adr,len,val$adr) boolean;
103 2 =      dcl (char$adr, val$adr, place) address,
    =      chars based char$adr (1) byte,
    =      value based val$adr address,
    =      (i,len) byte;
    =
104 2 =      value = 0;
105 2 =      place = 1;
106 2 =      do i = 1 to len;
107 3 =          if not number(chars(len - i)) then
108 3 =              return(false);
109 3 =              value = value + (chars(len - i) - '0') * place;
110 3 =              place = place * 10;
111 3 =      end;
112 2 =      return(true);
113 2 =      end make$numeric;
    =
114 1 =      set$vec: procedure(v$adr,num) public;
115 2 =      dcl v$adr address,          /* set bit number given by num */
    =      vector based v$adr address, /* 0 <= num <= 15          */
    =      num byte;
116 2 =      if num = 0 then
117 2 =          vector = vector or 1;
    =      else
118 2 =          vector = vector or shl(double(1),num);
119 2 =      end set$vec;
    =
120 1 =      bit$loc: procedure(vector) byte;
    =      /* return location of right most on bit vector */
121 2 =      dcl vector address,          /* 0 - 15          */
    =      i byte;
122 2 =      i = 0;
123 2 =      do while i < 16 and (vector and double(1)) = 0;
124 3 =          vector = shr(vector,1);
125 3 =          i = i + 1;
126 3 =      end;
127 2 =      return(i);
128 2 =      end bit$loc;
    =
129 1 =      get$nxt: procedure(vector$adr) byte;
130 2 =      dcl i byte,
    =      (vector$adr,mask) address,
    =      vector based vector$adr address;
    =      /*
    =          if debug then
    =              do; call print(.(cr,lf,'getnxt: vector = '));
    =              call pdecimal(vector,10000,false);
    =              end;
    =          */
131 2 =      if (i := bit$loc(vector)) > 15 then
132 2 =          return(0ffh);
133 2 =      mask = 1;
134 2 =      if i > 0 then
135 2 =          mask = shl(mask,i);

```

COPYRIGHT © 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

136 2 =      vector = vector xor mask;          /* turn off bit      */
      = /*
      =      if debug then
      =      do; call print(,(cr,lf,'getnxt: vector, i, mask $'));
      =      call pdecimal(vector,10000,false);
      =      call printb;
      =      call pdecimal(i,10000,false);
      =      call printb;
      =      call pdecimal(mask,10000,false);
      =      end;
      = /*
137 2 =      return(i);
138 2 =      end getnxt;          /* too bad plm rotates only work on byte values */
      =
      = /* help: procedure;      COMMENTED OUT - HELP PROGRAM REPLACE DISPLAY
      =
      =      call print(,(cr,lf,
      =      tab,tab,tab,'DIR EXAMPLES',cr,lf,lf,
      =      'dir file.one',tab,tab,tab,
      =      '(find a file on current user and default drive)',cr,lf,
      =      'dir *.com d:*.pli',tab,tab,'(find matching files on default and d: drive)',
      =      cr,lf,
      =      'dir [rw]',tab,tab,tab,'(find files that are read/write)',cr,lf,
      =      'dir [ro dir sys]',tab,tab,'(same for read/only, directory, system)',cr,lf,
      =      'dir [xpcb]',tab,tab,tab,'(find files with XFCB''s)',cr,lf,
      =      'dir [nonxpcb]',tab,tab,tab,'(find files without XFCB''s)',cr,lf,
      =      'dir [exclude] *.com',tab,tab,'(find files that don't end in ''com'')',cr,lf,
      =      'dir [nosort]',tab,tab,tab,'(don't sort the files)',cr,lf,
      =      'dir [full]',tab,tab,tab,'(show all file information)',cr,lf,
      =      'dir [size]',tab,tab,tab,'(show name and size in kilobytes)',cr,lf,
      =      'dir [short]',tab,tab,tab,'(show just the file names)',cr,lf,
      =      'dir [drive = all]',tab,tab,'(search all logged in drives)',cr,lf,
      =      'dir [drive = (a,b,p)]',tab,tab,
      =      '(search specified drives, ''disk'' is synonym)',cr,lf,
      =      'dir [user = all]',tab,tab,'(find files with any user number)',cr,lf,
      =      'dir [user = (0,1,15), G12]',tab,'(find files with specified user number)',
      =      cr,lf,
      =      'dir [length = n]',tab,tab,'(print headers every n lines)',cr,lf,
      =      'dir [ff]',tab,tab,tab,'(print form feeds between headers)',cr,lf,
      =      'dir [message user=all]',tab,tab,'(show user/drive areas with no files)',
      =      cr,lf,
      =      'dir [help]',tab,tab,tab,'(show this message)',cr,lf,
      =      'dir [dir sys rw ro sort xpcb nonxpcb full] d:*.*,tab,'(defaults$'));
      =
      =      call terminate;
      =      end help; */
      =
      = /* ----- Scanner Info ----- */
      =
      =      $include (scan.lit)
      =1
139 1 =1 declare
      =1      pcb$structure literally 'structure (
      =1      state address,
      =1      scan$adr address,
      =1      token$adr address,

```

COPYRIGHT © 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

      =1          tok$typ byte,
      =1          token$len byte,
      =1          p$level byte,
      =1          nxt$token byte)';
      =1
140  1 =1 declare
      =1          t$null lit '0',
      =1          t$param lit '1',
      =1          t$top lit '2',
      =1          t$mod lit '4',
      =1          t$identifier lit '8',
      =1          t$string lit '16',
      =1          t$numeric lit '32',
      =1          t$filespec lit '64',
      =1          t$error lit '128';
      =1
      =
141  1 = dcl pcb pcb$structure
      =          initial (0,,buff(0),,fcb,0,0,0,0) ;
      =
142  1 = dcl token based pcb.token$adr (12) byte;
143  1 = dcl got$options boolean;
      =
144  1 = get$options: procedure;
145  2 = dcl temp byte;
      =
146  2 = do while pcb.scan$adr <> 0ffffh and ((pcb.tok$typ and t$top) <> 0);
      =
147  3 = if pcb.nxt$token <> t$mod then do;
      =                                     /* options with no modifiers */
149  4 = if token(1) = 'A' then
150  4 = display$attributes = true;
      =
151  4 = else if token(1) = 'D' and token(2) = 'I' then
152  4 = find.dir = true;
      =
153  4 = else if token(1) = 'D' and token(2) = 'A' then do;
155  5 = format = form$full;
156  5 = date$opt = true;
157  5 = end;
      = /*
      =          else if token(1) = 'D' and token(2) = 'E' then
      =          debug = true;
      = */
158  4 = else if token(1) = 'E' then
159  4 = find.exclude = true;
      =
160  4 = else if token(1) = 'F' then do;
162  5 = if token(2) = 'F' then
163  5 = formfeeds = true;
164  5 = else if token(2) = 'U' then
165  5 = format = form$full;
166  5 = else goto op$error;
167  5 = end;
      =
168  4 = else if token(1) = 'G' then
169  4 = do;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

170 5 =      if pcb.token$len < 3 then
171 5 =          temp = token(2) - '0';
           =      else
172 5 =          temp = (token(2) - '0') * 10 + (token(3) - '0');
173 5 =          if temp >= 0 and temp <= 15 then
174 5 =              call set$vec(.usr$vector,temp);
175 5 =          else goto opterr;
176 5 =      end;
           =
           =      /* else if token(1) = 'H' then
           =          call help; */
           =
177 4 =      else if token(1) = 'M' then
178 4 =          message = true;
           =
179 4 =      else if token(1) = 'N' then
180 4 =          do;
181 5 =          if token(4) = 'X' then
182 5 =              find.nonxfcb = true;
183 5 =          else if token(3) = 'P' then
184 5 =              no$page$mode = OFFh;
185 5 =          else if token(3) = 'S' then
186 5 =              sort$op = false;
187 5 =          else goto opterr;
188 5 =      end;
           =
           =      /* else if token(1) = 'P' then
           =          find.pass = true; */
           =
189 4 =      else if token(1) = 'R' and token(2) = 'D' then
190 4 =          find.ro = true;
           =
191 4 =      else if token(1) = 'R' and token(2) = 'W' then
192 4 =          find.rw = true;
           =
193 4 =      else if token(1) = 'S' then do;
195 5 =          if token(2) = 'Y' then
196 5 =              find.sys = true;
197 5 =          else if token(2) = 'I' then
198 5 =              format = form$size;
199 5 =          else if token(2) = 'O' then
200 5 =              sort$op = true;
201 5 =          else goto opterr;
202 5 =      end;
           =
203 4 =      else if token(1) = 'X' then
204 4 =          find.xfcb = true;
           =
205 4 =      else goto opterr;
           =
206 4 =      call scan(.pcb);
207 4 =      end;
           =
           =      else
208 3 =          do;
209 4 =          if token(1) = 'L' then
210 4 =              do;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* options with modifiers */

```

211 5 =      call scan(.pcb);
212 5 =      if (pcb.tok$typ and t$numeric) <> 0 then
213 5 =          if make$numeric(.token(1),pcb.token$len,.page$len) then
214 5 =              if page$len < 5 then
215 5 =                  goto op$err;
216 5 =              else call scan(.pcb);
217 5 =              else goto op$err;
218 5 =          else goto op$err;
219 5 =      end;
      =
220 4 =      else if token(1) = 'U' then
221 4 =          do;
      =      /*
      =          if debug then
      =              call print(.(cr,lf,'In User option$'));
      =      */
222 5 =      call scan(.pcb);
223 5 =      if (((pcb.tok$typ and t$mod) = 0) or (bdos < bdos20)) then
224 5 =          goto op$err;
225 5 =      do while (pcb.tok$typ and t$mod) <> 0 and
      =          pcb.scan$adr <> 0ffffh;
226 6 =          if token(1) = 'A' and token(2) = 'L' then
227 6 =              usr$vector = 0ffffh;
228 6 =          else if (pcb.tok$typ and t$numeric) <> 0 and pcb.token$len < 3 then
229 6 =              do;
230 7 =                  if pcb.token$len = 1 then
231 7 =                      temp = token(1) - '0';
      =                  else
232 7 =                      temp = (token(1) - '0') * 10 + (token(2) - '0');
233 7 =                  if temp >= 0 and temp <= 15 then
234 7 =                      call set$vec(.usr$vector,temp);
235 7 =                  else goto op$err;
236 7 =                  end;
237 6 =          else goto op$err;
238 6 =          call scan(.pcb);
239 6 =      end;
240 5 =      end;      /* User option */
      =
241 4 =      else if token(1) = 'D' and (token(2) = 'R' or token(2) = 'I') then
242 4 =          do;      /* allow DRIVE or DISK */
243 5 =          call scan(.pcb);
244 5 =          if (pcb.tok$typ and t$mod) = 0 then
245 5 =              goto op$err;
246 5 =          do while (pcb.tok$typ and t$mod) <> 0 and
      =              pcb.scan$adr <> 0ffffh;
247 6 =              if token(1) = 'A' and token(2) = 'L' then
248 6 =                  do;
249 7 =                      drv$vector = 0ffffh;
250 7 =                      drv$vector = drv$vector and get$login;
251 7 =                  end;
252 6 =              else if token(1) >= 'A' and token(1) <= 'P' then
253 6 =                  call set$vec(.drv$vector,token(1) - 'A');
254 6 =              else goto op$err;
255 6 =              call scan(.pcb);
256 6 =          end;
257 5 =      end;      /* drive option */
      =

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

258 4 =      else goto op$err;
      =
259 4 =      end;                /* options with modifiers */
      =
260 3 =      end;      /* do while */
      =
261 2 =      got$options = true;
262 2 =      return;
      =
263 2 =      op$err;
      =      call print(('.ERROR: Illegal Option or Modifier.',
      =              cr,lf,'$'));
264 2 =      call terminate;
265 2 =      end got$options;
      =
266 1 =      get$file$spec: procedure;
267 2 =      dcl i byte;
268 2 =      if num$search$files < max$search$files then
269 2 =      do;
270 3 =          call move(f$namelen + f$typeLen,.token(1),
      =              ,search(num$search$files).name(0));
      =
271 3 =          if search(num$search$files).name(f$name - 1) = ' ' and
      =              search(num$search$files).name(f$type - 1) = ' ' then
272 3 =              search(num$search$files).anyfile = true; /* match on any file */
273 3 =          else search(num$search$files).anyfile = false; /* speedier compare */
      =
274 3 =          if token(0) = 0 then
275 3 =              search(num$search$files).drv = 0ffh; /* no drive letter with */
      =              else /* file spec- */ /*
276 3 =              search(num$search$files).drv = token(0) - 1;
      =              /* 0ffh in drv field indicates to look on all drives that will be */
      =              /* scanned as set by the 'drive =' option, see 'match:' proc in */
      =              /* search.plm module */ /*
      =
277 3 =          num$search$files = num$search$files + 1;
278 3 =      end;
      =      else
279 2 =      do; call print(('.File Spec Limit is $'));
281 3 =          call p$decimal(max$search$files,100,true);
282 3 =          call crlf;
283 3 =      end;
284 2 =      call scan(.pcb);
285 2 =      end get$file$spec;
      =
286 1 =      set$defaults: procedure;
      =          /* set defaults if not explicitly set by user */
287 2 =          if not (find.dir or find.sys) then
288 2 =              find.dir, find.sys = true;
289 2 =          if not (find.ro or find.rw) then
290 2 =              find.rw, find.ro = true;
      =
291 2 =          if find.xfcb or find.nonxfcb then
292 2 =              do; if format = form$short then
294 3 =                  format = form$full;
295 3 =              end;
      =          else /* both xfcb and nonxfcb are off */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

296 2 =      find.nonxfcb, find.xfcb = true;
      =
297 2 =      if num$search$files = 0 then
298 2 =      do;
299 3 =          search(num$search$files).anyfile = true;
300 3 =          search(num$search$files).drv = Offh;
301 3 =          num$search$files = 1;
302 3 =      end;
      =
303 2 =      if drv$vector = 0 then
304 2 =          do i = 0 to num$search$files - 1;
305 3 =              if search(i).drv = Offh then search(i).drv = cur$drv;
307 3 =              call set$vec(.drv$vector,search(i).drv);
308 3 =          end;
      =
      else /* a '[drive =' option was found */
309 2 =          do i = 0 to num$search$files - 1;
310 3 =              if search(i).drv <> Offh and search(i).drv <> cur$drv then
311 3 =              do; call print(('ERROR: Illegal Global/Local ',
      =                  'Drive Spec Mixing.',cr,lf,'$'));
313 4 =              call terminate;
314 4 =          end;
315 3 =      end;
316 2 =      if usr$vector = 0 then
317 2 =          call set$vec(.usr$vector,get$usr);
      =
      = /* set up default page size for display */
318 2 =      if bdos > bdos30 then do;
320 3 =          if not formfeeds then do;
322 4 =              if page$len = Offfffh then do;
324 5 =                  page$len = get$cbbyte(page$len$offset);
325 5 =                  if page$len < 5 then
326 5 =                      page$len = 24;
327 5 =              end;
328 4 =          end;
329 3 =      end;
330 2 =      end set$defaults;
      =
331 1 =      dcl (save$uvec,temp) address;
332 1 =      dcl i byte;
333 1 =      declare last$dseq$byte byte
      =          initial (0);
      =
334 1 =      pla;
      =          do;
335 2 =              os = high(get$version);
336 2 =              bdos = low(get$version);
      =
337 2 =              if bdos < bdos30 or os = mpm then do;
339 3 =                  call print(('Requires CP/M 3',cr,lf,'$'));
340 3 =                  call terminate; /* check to make sure function call is valid */
341 3 =              end;
      =          else
342 2 =              call set$console$mode;
      =
      = /* note - initialized declarations set defaults */
343 2 =      cur$drv = get$cur$drv;
344 2 =      call scan$init(.pcb);

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

345 2 = call scan(,pcb);
346 2 = no$page$mode = getscbbyte(nopage$mode$offset);
347 2 = got$options = false;
348 2 = do while pcb.scan$adr <> 0ffffh;
349 3 =     if (pcb.tok$typ and t$op) <> 0 then
350 3 =         if got$options = false then
351 3 =             call get$options;
352 3 =         else
353 4 =             do;
354 4 =                 call print(,('ERROR: Options not grouped together.',
355 4 =                     cr,lf,'$'));
356 3 =                 call terminate;
357 3 =             end;
358 3 =         else if (pcb.tok$typ and t$filespec) <> 0 then
359 4 =             call get$file$spec;
360 4 =         else
361 4 =             do;
362 3 =                 call print(,('ERROR: Illegal command tail.',cr,lf,'$'));
363 2 =                 call terminate;
364 2 =             end;
365 2 =         end;
366 2 =     end;
367 2 = call set$defaults;
368 2 =
369 2 = /* main control loop */
370 2 =
371 2 = call search$init; /* set up memory pointers for subsequent storage */
372 2 =
373 2 = do while (cur$drv := get$next(,drv$vector)) <> 0fffh;
374 3 =     call select$drive(cur$drv);
375 3 =     save$uvec = usr$vector; /* user numbers to search on each drive */
376 3 =     active$usr$vector = 0; /* users active on cur$drv */
377 3 =     cur$usr = get$next(,usr$vector); /* get first user num and mask */
378 3 =     get$all$dir$entries = false; /* off it off */
379 3 =     if usr$vector <> 0 and format <> form$short then
380 4 =         do; /* find high water mark if */
381 4 =             fcb(f$drvusr) = '?'; /* more than one user requested */
382 4 =             i = search$first(,fcb); /* get first directory entry */
383 4 =             temp = 0;
384 4 =             do while i <> 255;
385 5 =                 temp = temp + 1;
386 5 =                 i = search$next;
387 5 =             end; /* is there enough space in the */
388 4 =             if maxb > mult23(temp) + shl(temp,1) then /* worst case ? */
389 4 =                 get$all$dir$entries = true; /* location of last possible */
390 4 =             end; /* file info record and add */
391 3 =             first$pass = true; /* room for sort indices */
392 3 =             active$usr$vector = 0ffffh;
393 3 =             do while cur$usr <> 0fffh;
394 4 =                 if debug then
395 4 =                     call print(,('in user loop $'));
396 4 =                 call set$vec(,temp,cur$usr);
397 4 =                 if (temp and active$usr$vector) <> 0 then

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

388 4 =          do;
389 5 =          if format <> form$short and
              (first$pass or not get$all$dir$entries) then
390 5 =          do;
391 6 =          call get$files;      /* collect files in memory and */
392 6 =          first$pass = false; /* build the active usr vector */
393 6 =          sorted = false;     /* sort module will set sorted */
394 6 =          if sort$op then     /* to true, if successful sort */
395 6 =              call sort;
396 6 =          end;
397 5 =          call display$files;
398 5 =          end;
399 4 =          cur$usr = get$next(.usr$vector);
400 4 =          end;
401 3 =          usr$vector = save$uvec;      /* restore user vector for nxt */
402 3 =          end; /* do while drv$usr;    drive scan      */
              =
              =
403 2 =          if not file$displayed and not message then
404 2 =              call print(,('No File',cr,lf,'$'));
405 2 =          call terminate;
              =
406 2 =          end;
407 1 =          end sdir;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0845H   2885D
VARIABLE AREA SIZE = 01CEH   462D
MAXIMUM STACK SIZE = 0008H    8D
760 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DISPLAY

OBJECT MODULE PLACED IN DISP

COMPILER INVOKED BY: PLM80 DISP,PLM DEBUG PAGEWIDTH(130) OPTIMIZE OBJECT(DISPLAY)

```

1      $title ('SDIR - Display Files')
      display;
      do;

          /* Display Module for SDIR */

          $include(comlit.lit)

          =
2      1 = declare
          =      lit      literally      'literally',
          =      dcl      lit      'declare',
          =      true      lit      'offh',
          =      false      lit      '0',
          =      boolean      lit      'byte',
          =      forever      lit      'while true',
          =      cr      lit      '13',
          =      lf      lit      '10',
          =      tab      lit      '9',
          =      ctrlc      lit      '3',
          =      ff      lit      '12',
          =      page$len$offset      lit      '1ch',
          =      nopage$mode$offset      lit      '2Ch',
          =      sectorlen      lit      '128';

          $include(mon.plm)

          =
          =      /* definitions for assembly interface module      */
3      1 = declare
          =      fcb (33) byte external,      /* default file control block      */
          =      maxb address external,      /* top of memory      */
          =      buff(128) byte external;      /* default buffer      */
          =

          4      1 = mon1: procedure(f,a) external;
          5      2 =      declare f byte, a address;
          6      2 =      end mon1;
          =

          7      1 = mon2: procedure(f,a) byte external;
          8      2 =      declare f byte, a address;
          9      2 =      end mon2;
          =

          10     1 = mon3: procedure(f,a) address external;
          11     2 =      declare f byte, a address;
          12     2 =      end mon3;
          =

          13     1 = dcl debug boolean external;
          14     1 = dcl (cur$drv, cur$usr) byte external;

          15     1 = dcl (os,bdos) byte external;
          $include(vers.lit)

          16     1 = declare
          =      bdos20 lit '20h',

```

CP/M Plus V3.0

COPYRIGHT © 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # *CP3-135*

```

=      bdos30  lit '30h',
=      mpw     lit '01h',
=      mpw86   lit '11h';

17  1      dcl used$de address external;      /* number of used directory entries */
18  1      dcl date$opt boolean external;     /* date option flag */
19  1      dcl display$attributes boolean external; /* attributes display flag */
20  1      dcl sorted boolean external;
21  1      dcl filesfound address external;
22  1      dcl no$page$mode byte external;
23  1      dcl sfcbs$present byte external;   /* sfcbs there/not there indicator */

      $include (search.lit)

=
24  1  =  declare                      /* what kind of file user wants to find      */
=      find$structure lit 'structure (
=      dir byte,
=      sys byte,
=      ro byte,
=      rw byte,
=      pass byte,
=      xfcbs byte,
=      nonxfcb byte,
=      exclude byte)';
=
25  1  =  declare
=      max$search$files literally '10';
=
26  1  =  declare
=      search$structure lit 'structure(
=      drv byte,
=      name(8) byte,
=      type(3) byte,
=      anyfile boolean)';      /* match on any drive if true */
=
27  1      dcl find find$structure external;

28  1      dcl format byte external,          /* format is one of the following */
=      page$len address external, /* page size before printing new headers */
=      message boolean external, /* print titles and msg when no file found */
=      formfeeds boolean external; /* use form feeds to separate headers */

      $include(format.lit)

=
29  1  =  dcl form$short lit '0',      /* format values for SDIR */
=      form$size lit '1',
=      form$full lit '2';
=

30  1      dcl file$displayed boolean public initial (false);
=      /* true if we ever display a file, from any drive or user */
=      /* used by main.plm for file not found message */

31  1      dcl dir$label byte external;

      $include(fcb.lit)
=

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```

32 1 = declare
    = f$drvusr      lit '0',      /* drive/user byte      */
    = f$name        lit '1',      /* file name            */
    = f$namelen     lit '8',      /* file name length     */
    = f$type        lit '9',      /* file type field      */
    = f$typelen     lit '3',      /* type length          */
    = f$rw          lit '9',      /* high bit is R/W attribute */
    = f$dirsyst     lit '10',     /* high bit is dir/sys attribute */
    = f$arc         lit '11',     /* high bit is archive attribute */
    = f$ex          lit '12',     /* extent               */
    = f$s1          lit '13',     /* module byte          */
    = f$rc          lit '15',     /* record count         */
    = f$diskmap     lit '16',     /* file disk map        */
    = diskmaplen    lit '16',     /* disk map length      */
    = f$drvusr2     lit '16',     /* fcb2                 */
    = f$name2       lit '17',
    = f$type2       lit '25',
    = f$rrrec       lit '33',     /* random record        */
    = f$rrco        lit '35';    /* ' ' overflow        */
    =
    $include(xfcb.lit)

33 1 = declare                                /* XFCB                */
    = xfc$type      lit '10h',    /* identifier on disk   */
    = xfc$passmode  lit '12',    /* pass word protection mode */
    = xfc$pass      lit '16',    /* XFCB password        */
    = passlen       lit '8',      /* password length      */
    = xfc$create    lit '24',    /* creation/access time stamp */
    = xfc$update    lit '28';    /* update time stamp    */
    =

34 1 = declare                                /* directory label: special case of XFCB */
    = dirlabeltype  lit '20h',    /* identifier on disk   */
    = dl$password   lit '128',    /* masks on data byte   */
    = dl$access     lit '64',
    = dl$update     lit '32',
    = dl$makexfcb   lit '16',
    = dl$exists     lit '1';

35 1 = declare                                /* password mode of xfcb */
    = pm$read       lit '80h',
    = pm$write      lit '40h',
    = pm$delete     lit '20h';

36 1 dcl
    buf$fc$adr address external,      /* index into directory buffer */
    buf$fc based buf$fc$adr (32) byte,
                                         /* fcb template for dir      */

    (f$i$adr,last$f$i$adr,first$f$i$adr) address external,
    cur$file address;                  /* number of file currently   */
                                         /* being displayed            */

    $include(finfo.lit)

    =
    = /* file info record for SDIR - note if this structure changes in size */
    = /* the multXX: routine in the sort.plm module must also change */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

```

63  2      end add3byte;          /* add word to 3 byte structure */

64  1      add3byte3: procedure (byte3$adr,byte3) external;
65  2          dcl (byte3$adr, byte3) address;
66  2      end add3byte3;          /* add 3 byte quantity to 3 byte total */

67  1      shr3byte: procedure (byte3$adr) external;
68  2          dcl byte3$adr address;
69  2      end shr3byte;

/* ----- Routines in search.plm ----- */

70  1      search$first: procedure(fcb$adr) byte external;
71  2          dcl fcb$adr address;
72  2      end search$first;

73  1      search$next: procedure byte external;
74  2      end search$next;

75  1      break: procedure external;
76  2      end break;

77  1      match: procedure boolean external;
78  2          dcl fcb$adr address;
79  2      end match;

/* ----- Other external routines ----- */

80  1      display$time$stamp: procedure (ts$adr) external;    /* in dts.plm */
81  2          dcl ts$adr address;
82  2      end display$time$stamp;

83  1      terminate: procedure external;                      /* in main.plm */
84  2      end terminate;

85  1      mult23: procedure(index) address external;          /* in sort.plm */
86  2          dcl index address;
87  2      end mult23;

/* ----- From dpb86.plm or dpb80.plm ----- */

#include(dpb.lit)

=
= /* indices into disk parameter block, used as parameters to dpb procedure */
=
1 = dcl      spt$w      lit      '0',
=          blkshf$b    lit      '2',
=          blkmsk$b    lit      '3',
=          extmsk$b    lit      '4',
=          blkmax$w    lit      '5',
=          dirmax$w    lit      '7',
=          dirblk$w    lit      '9',
=          chksiz      lit      '11',
=          offset$w    lit      '13';

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

      =

89  1      dpb$byte; procedure (dpb$index) byte external;
90  2          dcl dpb$index byte;
91  2      end dpb$byte;

92  1      dpb$word; procedure (dpb$index) address external;
93  2          dcl dpb$index byte;
94  2      end dpb$word;

/* ----- routines and data structures local to this module ----- */

95  1      direct$console$io; procedure byte;
96  2          return mon2(6,0ffh); /* ff to stay downward compatible */
97  2      end direct$console$io;

98  1      dcl first$time address initial (0);

/*- - - - - */

99  1      wait$keypress; procedure;
100 2          declare char byte;
/* if debug then
call print(.(cr,lf,'In wait$keypress...',cr,lf,'$'));
*/
101 2          char = direct$console$io;
102 2          do while char = 0;
103 3              char = direct$console$io;
104 3          end;
105 2          if char = ctrlc then
106 2              call terminate;
107 2          end wait$keypress;

108 1      declare global$line$count byte initial(1);

/*- - - - - */

109 1      crlf$and$check; procedure;
/*
if debug then
call print(.(cr,lf,'In crlf$and$check...',cr,lf,'$'));
*/
110 2          if no$page$mode = 0 then do;
112 3              if global$line$count > page$len-1 then do;
114 4                  call print(.(cr,lf,'Press RETURN to Continue $'));
115 4                  cur$line = cur$line + 1;
116 4                  call wait$keypress;
117 4                  global$line$count = 0;
118 4              end; /* global$line$count > page$len */
119 3          end; /* no$page$mode = 0 */
120 2          call crlf;
121 2          global$line$count = global$line$count + 1;
122 2          end crlf$and$check;

123 1      dcl      total$kbytes structure ( /* grand total k bytes of files matched */
          lword address,

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

        hbyte byte),
        total$recs structure (      /* grand total records of files matched */
            lword address,
            hbyte byte),
        total$1k$blocks structure( /* how many 1k blocks are allocated   */
            lword address,
            hbyte byte);

/*-----*/

124  1  add$totals: procedure;

/*
    if debug then
call print(.(cr,lf,'In add$totals...',cr,lf,'$'));
*/

125  2      call add3byte(.total$kbytes,file$info.kbytes);
126  2      call add3byte3(.total$recs,.file$info.recs$lword); /* records in file */
127  2      call add3byte(.total$1k$blocks,file$info.onekblocks);

128  2  end add$totals;

129  1  dcl files$per$line byte;
130  1  dcl cur$line address;

131  1  dcl hdr (*) byte data      ('   Name   Bytes  Recs  Attributes $');
132  1  dcl hdr$bars (*) byte data ('-----');
133  1  dcl hdr$pu (*) byte data   ('  Prot   Update  $');
134  1  dcl hdr$xfcb$bars (*) byte data ('-----');
135  1  dcl hdr$access (*) byte data      ('   Access  $');
136  1  dcl hdr$create (*) byte data      ('   Create  $');

                                /* example date      04/02/55 00:34 */

/*-----*/

137  1  display$file$info: procedure;

                                /* print filename.typ */

/*
    if debug then
call print(.(cr,lf,'In display$file$info...',cr,lf,'$'));
*/

138  2      call printfn(.file$info.name(0));
139  2      call printb;
140  2      call pdecimal(file$info.kbytes,10000,true);
141  2      call printchar('K');                                /* up to 32 Meg - Bytes */
                                                                /* or 32,000K          */

142  2      call printb;
143  2      call p3byte(.file$info.recs$lword,1);              /* records              */
144  2      call printb;
145  2      if rol(file$info.name(f$dirsys-1),1) then          /* Type                  */
146  2          call print(.( 'Sys$'));
147  2      else call print(.( 'Dir$'));
148  2      call printb;
149  2      if rol(file$info.name(f$rw-1),1) then
150  2          call print(.( 'RO$'));
151  2      else call print(.( 'RW$'));
152  2      call printb;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

153 2      if not display$attributes then do;
155 3          if rol(file$info.name(f$arc-1),1) then
156 3              call print(('Arcv $'));
              else
157 3                  call print(('      $'));
158 3      end;
159 2      else do;
160 3          if rol(file$info.name(f$arc-1),1) then          /* arc bit was on in all */
161 3              call print$char('A');                      /* dir entries          */
162 3          else call printb;
163 3          if rol(file$info.name(0),1) then
164 3              call print$char('1');
165 3          else call printb;
166 3          if rol(file$info.name(1),1) then
167 3              call print$char('2');
168 3          else call printb;
169 3          if rol(file$info.name(2),1) then
170 3              call print$char('3');
171 3          else call printb;
172 3          if rol(file$info.name(3),1) then
173 3              call print$char('4');
174 3          else call printb;
175 3      end;
176 2      end display$file$info;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

/*-----*/

```

177 1      display$xfcb$info: procedure;
/*
if debug then
call print((',cr,lf,'In display$xfcb$info...',cr,lf,$'));
*/
178 2          if file$info.x$i$adr < 0 then
179 2              do;
180 3                  call printb;
181 3                  x$i$adr = file$info.x$i$adr;
182 3                  if (xfcb$info.passmode and pm$read) < 0 then
183 3                      call print(('Read $'));
184 3                  else if (xfcb$info.passmode and pm$write) < 0 then
185 3                      call print(('Write $'));
186 3                  else if (xfcb$info.passmode and pm$delete) < 0 then
187 3                      call print(('Delete$'));
                  else
188 3                      call print(('None $'));
189 3                  call printb;
190 3                  if (xfcb$info.update(0) < 0 or xfc$b$info.update(1) < 0) then
191 3                      call display$timestamp(xfc$b$info.update);
192 3                  else call print(('      $'));
193 3                  call printb; call printb;
194 3                  if (xfcb$info.create(0) < 0 or xfc$b$info.create(1) < 0) then
195 3                      call display$timestamp(xfc$b$info.create(0));
196 3                      /* Create/Access */

197 3          end;
198 2      end display$xfcb$info;

199 1      dcl first$title boolean initial (true);

```

```

/*-----*/

200 1  display$title: procedure;
/*
   if debug then
   call print((cr,lf,'In display$title...',cr,lf,'$'));
*/

201 2      if formfeeds then
202 2          call print$char(ff);
203 2      else if not first$title then
204 2          call crlf$and$check;
   call print(('Directory For Drive $'));
206 2      call printchar('A'+ cur$drv); call printchar(':');
208 2      if bdos >= bdos20 then
209 2          do;
210 3              call print((' User $'));
211 3              call pdecimal(cur$usr,10,true);
212 3          end;
213 2          call crlf$and$check;
214 2          cur$line = 2;
215 2          first$title = false;
216 2      end display$title;

/*-----*/

217 1  short$display: procedure (fname$adr);
218 2      dcl fname$adr address;
/*
   if debug then
   call print((cr,lf,'In short$display...',cr,lf,'$'));
*/

219 2      if cur$file mod files$per$line = 0 then
220 2          do;
221 3          if cur$line mod page$len = 0 and first$time = 0 then
222 3              do;
223 4                  call crlf$and$check;
224 4                  call display$title;
225 4                  call crlf$and$check;
226 4              end;
227 3          else
228 3              call crlf$and$check;
229 3              cur$line = cur$line + 1;
230 3              call printchar(cur$drv + 'A');
231 3          end;
232 2      else call printb;
233 2      call print((': $'));
234 2      call printfn(fname$adr);
235 2      call break;
236 2      cur$file = cur$file + 1;
237 2      first$time = first$time + 1;
   end short$display;

/*-----*/

238 1  test$att: procedure(char,off,on) boolean;
239 2      dcl (char,off,on) byte;
/*

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

        if debug then
        call print(,(cr,lf,'In test$att...',cr,lf,'$'));
        */
240 2      if (80h and char) <> 80h and off then
241 2          return(true);
242 2      if (80h and char) = 80h and on then
243 2          return(true);
244 2      return(false);
245 2  end test$att;

        /*-----*/

246 1  right$attributes: procedure(name$adr) boolean;
247 2      dcl name$adr address,
        name based name$adr (1) byte;
248 2      return
        test$att(name(f$rw-1),find.rw,find.ro) and
        test$att(name(f$dirs-1),find.dir,find.sys);
249 2  end right$attributes;

        /*-----*/

250 1  short$dir: procedure;          /* looks like 'DIR' command */
251 2      dcl dcnt byte;
        /*
        if debug then
        call print(,(cr,lf,'In short$dir...',cr,lf,'$'));
        */
252 2      fcb(f$drvusr) = '?';
253 2      files$per$line = 4;
254 2      dcnt = search$first(.fcb);
255 2      do while dcnt <> 0ffh;
256 3          buf$fcb$adr = shl(dcnt and 11b,5)+.buff;    /* dcnt mod 4 * 32    */
257 3          if (buf$fcb(f$drvusr) and 0f0h) = 0 and
                buf$fcb(f$ex) = 0 and
                buf$fcb(f$ex) <= dpb$byte(extmsk$b) then /* no dir labels, xfcbs */
258 3              if match then
259 3                  if right$attributes(.buf$fcb(f$name)) then
260 3                      call short$display(.buf$fcb(f$name));
261 3                  dcnt = search$next;
262 3              end;
263 2  end short$dir;

264 1  dcl (last$plus$one,index) address;

        /*-----*/

265 1  getnxt$file$info: procedure;    /* set f$i$adr to base file$info on file  */
266 2      dcl right$usr boolean;      /* to be displayed, f$i$adr = 0ffffh if end */
        /*
        if debug then
        call print(,(cr,lf,'In getnxt$file$info...',cr,lf,'$'));
        */
267 2      right$usr = false;
268 2      if sorted then
269 2          do; index = index + 1;
271 3          f$i$adr = mult23(f$i$indices(index));

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93050

SER. # _____


```

272 3      do while file$info.usr < cur$usr and index < filesfound;
273 4          index = index + 1;
274 4          f$i$adr = mult23(f$i$indices(index));
275 4      end;
276 3      if index = files$found then
277 3          f$i$adr = last$plus$one;          /* no more files */
278 3      end;
      else /* not sorted display in order found in directory */
279 2      do; /* use last$plus$one to avoid wrap around problems */
280 3          f$i$adr = f$i$adr + size(file$info);
281 3          do while file$info.usr < cur$usr and f$i$adr < last$plus$one;
282 4              f$i$adr = f$i$adr + size(file$info);
283 4          end;
284 3      end;
285 2      end getnxt$file$info;

```

```
/*-----*/
```

```

286 1      size$display: procedure;
      /*
      if debug then
      call print(,cr,lf,'In size$display...',cr,lf,'$');
      */
287 2      if (format and form$size) < 0 then
288 2          files$per$line = 3;
289 2      else files$per$line = 4;
290 2      do while f$i$adr < last$plus$one;
291 3      if ((file$info.x$i$adr < 0 and find.xfcb) or
          file$info.x$i$adr = 0 and find.nonxfcb) and
          right$attributes(.file$info.name(0)) then
292 3          do;
293 4              call add$totals;
294 4              call short$display(.file$info.name(0));
295 4              call pdecimal(file$info.kbytes,10000,true);
296 4              call print(,('k$'));
297 4          end;
298 3          call getnxt$file$info;
299 3      end;
300 2      end size$display;

```

```
/*-----*/
```

```

301 1      display$no$dirlabel: procedure;
      /*
      if debug then
      call print(,cr,lf,'In display$no$dirlabel...',cr,lf,'$');
      */
302 2      files$per$line = 2;
303 2      first$time = 0;
304 2      do while (f$i$adr < last$plus$one);
305 3      if ((file$info.x$i$adr < 0 and find.xfcb) or
          (file$info.x$i$adr = 0 and find.nonxfcb)) and
          right$attributes(.file$info.name(0)) then
306 3      do;
307 4      if ((cur$file mod files$per$line) = 0) then /* need new line */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

308 4      do;

309 5          if ((cur$line mod page$len) = 0) then
310 5      do;

311 6          if ((no$page$mode = 0) or (first$time = 0)) then do;
313 7              call crlf$and$check;
314 7              call display$title;
315 7              call crlf$and$check;
316 7              call print(,hdr);
317 7              call printb;                      /* two sets of hdrs */
318 7              call print(,hdr);
319 7              call crlf$and$check;
320 7              call print(,hdr$bars);
321 7              call printb;
322 7              call print(,hdr$bars);
323 7              call crlf$and$check;
324 7              cur$line = cur$line + 4;
325 7              first$time = first$time+1;
326 7          end;
327 6      else do;
328 7          call crlf$and$check;
329 7          cur$line = cur$line + 1;
330 7      end; /* no$page$mode check */

331 6      end;
332 5      else
333 5      do; call crlf$and$check;
334 6          cur$line = cur$line + 1;
335 6      end;

336 5      end;
337 4      else
338 4          call printb;                      /* separate the files */

338 4          call display$file$info;
339 4          cur$file = cur$file + 1;
340 4          call add$totals;
341 4          call break;
342 4      end;
343 3      call getnxt$file$info;
344 3      end;

345 2      end display$no$dirlabel;

/*-----*/

346 1      display$with$dirlabel: procedure;
347 1      /*
348 1      if debug then
349 1      call print(,(cr,lf,'In display$with$dirlabel...',cr,lf,'$'));
350 1      */

347 2          files$per$line = 1;
348 2          first$time = 0;
349 2          do while (f$i$adr < last$plus$one);

350 3          if ((file$info.x$i$adr < 0 and find.xfcb) or

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

      (file$info.%i$adr = 0 and find.nonxfcb)) and
      right$attributes(.file$info.name(0)) then
351  3      do;

352  4          if cur$line mod page$len = 0 then
353  4          do;

354  5              if ((no$page$mode = 0) or (first$time = 0)) then do;

356  6                  call crlf$and$check;
357  6                  call display$title;
358  6                  call crlf$and$check;
359  6                  call print(.hdr);
360  6                  call print(.hdr$pu);
361  6                  if (dirlabel and dl$access) < 0 then
362  6                      call print(.hdr$access);
                      else
363  6                          call print(.hdr$create);
364  6                          call crlf$and$check;
365  6                          call print(.hdr$bars);
366  6                          call print(.hdr%xfcb$bars);
367  6                          call crlf$and$check;
368  6                          cur$line = cur$line + 4;
369  6                          first$time = first$time + 1;
370  6                          end; /* no$page$mode check */

371  5          end;

372  4          call crlf$and$check;
373  4          cur$line = cur$line + 1;
374  4          call display$file$info;          /* display non bdos 3.0 file info */
375  4          call display%xfcb$info;
376  4          cur$file = cur$file + 1;
377  4          call break;
378  4          call add$totals;
379  4          end;
380  3          call getnxt$file$info;
381  3          end;
382  2      end display$with$dirlabel;

/*- - - -MAIN ENTRY POINT - - - - -*/

383  1      display$files: procedure public; /* MODULE ENTRY POINT */
                                     /* display the collected data */
/*
if debug then
call print(.(cr,lf,'In main display routine...',cr,lf,'$'));
*/
384  2          cur$line, cur$file = 0;          /* force titles and new line */
385  2          total$bytes.lword, total$bytes.hbyte, total$recs.lword, total$recs.hbyte = 0;
386  2          total$1k$blocks.lword, total$1k$blocks.hbyte = 0;
387  2          f$i$adr = first$f$i$adr - size(file$info);          /* initial if no sort */
388  2          last$plus$one = last$f$i$adr + size(file$info);
389  2          index = 0ffffh;          /* initial if sorted */
390  2          call getnxt$file$info;          /* base file info record */

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

391 2      if format > 2 then
392 2      do;
393 3          call print(,('ERROR: Illegal Format Value.',cr,lf,'$'));
394 3          call terminate;          /* default could be patched - watch it */
395 3      end;

396 2      do case format;          /* format = */
397 3          call short$dir;          /* form$short */
398 3          call size$display;      /* form$size */
                                   /* form = full */

399 3          if date$opt then do;
401 4              if ((( dir$label and dl$exists) <> 0 ) and
                  ((( dir$label and dl$access) <> 0 ) or
                  (( dir$label and dl$update) <> 0 ) or
                  (( dir$label and dl$make$fcbs) <> 0 )) and (sfcb$present)) then
402 4                  call display$with$dirlabel; /* Timestamping is active! */
403 4              else do;
404 5                  call print(,('ERROR: Date and Time Stamping Inactive.',cr,lf,'$'));
405 5                  call terminate;
406 5              end;
407 4          end;
408 3          else do; /* No date option; Regular Full display */
409 4              if (((dir$label and dl$exists) <> 0) and (sfcb$present)) then
410 4                  do;
411 5                      call display$with$dirlabel;
412 5                  end;
413 4                  else
414 5                      do;
415 5                          call display$no$dirlabel;
416 4                      end;
417 3                  end; /* end of case */
418 2          if format <> form$short and cur$file > 0 then /* print totals */
419 2          do;
420 3              if cur$line + 4 > page$len and form$feeds then
421 3              do;
422 4                  call printchar(cr);
423 4                  call printchar(ff);          /* need a new page ? */
424 4              end;
425 3              else
426 4              do;
427 4                  call crlf$and$check;
428 4                  call crlf$and$check;
429 3              end;
430 3              call print(,('      Total Bytes      = $'));
431 3              call p3byte(,total$Kbytes,1);    /* 6 digit max */
432 3              call printchar('K');
433 3              call print(,(' Total Records = $'));
434 3              call p3byte(,total$recs,10);     /* 7 digit max */
435 3              call print(,(' Files Found = $'));
436 3              call pdecimal(cur$file,1000,true); /* 4 digit max */
437 3              call print(,('Total 1K Blocks = $'));
438 3              call p3byte(,total$1K$blocks,1); /* 6 digit max */
439 3              call print(,(' Used/Max Dir Entries For Drive $'));
440 3              call print$char('A' + cur$drv);
                  call printb;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

PL/M-80 COMPILER SDIR - DISPLAY FILES

```

442 3      call pdecimal(used$de,1000,true);
443 3      call print$char('/');
444 3      call pdecimal(dpb$word(dirmax$w) + 1,1000,true);
445 3      end;

446 2      if cur$file = 0 then
447 2      do;
448 3          if message then
449 3          do; call crlf$and$check;
451 4              call display$title;
452 4              call print(.( 'No File',crlf,$' ));
453 4          end;
454 3          call break;
455 3      end;
456 2      else do;
457 3          file$displayed = true;
458 3          if not formfeeds then
459 3          call crlf$and$check;
460 3      end;

461 2      end display$files;

462 1      end display;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

MODULE INFORMATION:

CODE AREA SIZE = 0B31H 2865D
 VARIABLE AREA SIZE = 0023H 35D
 MAXIMUM STACK SIZE = 000EH 14D
 815 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DPB80

OBJECT MODULE PLACED IN DPB80

COMPILER INVOKED BY: PLM80 DPB80.PLN DEBUG PAGESWIDTH(130) OPTIMIZE OBJECT(DPB80)

```

1      $title ('SDIR 8080 - Get Disk Parameters')
      dpb80:
      do;
          /* the purpose of this module is to allow independence */
          /* of processor, i.e., 8080 or 8086                      */

      $include (comlit.lit)

      =
2      1 = declare
      =
      =      lit      literally      'literally',
      =      dcl      lit      'declare',
      =      true      lit      'Offh',
      =      false      lit      '0',
      =      boolean      lit      'byte',
      =      forever      lit      'while true',
      =      cr      lit      '13',
      =      lf      lit      '10',
      =      tab      lit      '9',
      =      ctrlc      lit      '3',
      =      ff      lit      '12',
      =      page$len$offset      lit      '1ch',
      =      nopage$mode$offset      lit      '2Ch',
      =      sectorlen      lit      '128';

      /* function call 32 in 2.0 or later BDOS, returns the address of the disk
      parameter block for the currently selected disk, which consists of:
          spt      (2 bytes) number of sectors per track
          blkshf      (1 byte) block size = shl(double(128),blkshf)
          blkmsk      (1 byte) sector# and blkmsk = block number
          extmsk      (1 byte) logical/physical extents
          blkmax      (2 bytes) max alloc number
          dirmax      (2 bytes) size of directory-1
          dirblk      (2 bytes) reservation bits for directory
          chksiz      (2 bytes) size of checksum vector
          offset      (2 bytes) offset for operating system

      */

      $include(dpb.lit)

      =
      =      /* indices into disk parameter block, used as parameters to dpb procedure */
      =
3      1 = dcl      spt$w      lit      '0',
      =      blkshf$b      lit      '2',
      =      blkmsk$b      lit      '3',
      =      extmsk$b      lit      '4',
      =      blkmax$w      lit      '5',
      =      dirmax$w      lit      '7',
      =      dirblk$w      lit      '9',
      =      chksiz      lit      '11',
      =      offset$w      lit      '13';
      =

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```

        $include(mon.plm)
=
=          /* definitions for assembly interface module      */
4  1  =  declare
=          fcb (33) byte external,      /* default file control block */
=          maxb address external,      /* top of memory              */
=          buff(128)byte external;      /* default buffer             */
=
5  1  =  mon1: procedure(f,a) external;
6  2  =      declare f byte, a address;
7  2  =      end mon1;
=
8  1  =  mon2: procedure(f,a) byte external;
9  2  =      declare f byte, a address;
10 2  =      end mon2;
=
11 1  =  mon3: procedure(f,a) address external;
12 2  =      declare f byte, a address;
13 2  =      end mon3;
=
14 1  declare k$per$block address public;
15 1  declare dpb$base address;
16 1  declare dpb$array based dpb$base (15) byte;

17 1  dcl get$dpb lit '31';

18 1  dpb$byte: procedure(param) byte public;
19 2      dcl param byte;
20 2      return(dpb$array(param));
21 2  end dpb$byte;

22 1  dpb$word: procedure(param) address public;
23 2      dcl param byte;
24 2      return(dpb$array(param) + shl(double(dpb$array(param+1)),8));
25 2  end dpb$word;

26 1  base$dpb: procedure public;
27 2      dpb$base = mon3(get$dpb,0);
28 2      k$per$block = shr(dpb$byte(blkmsk$b)+1,3);
29 2  end base$dpb;

30 1  end dpb80;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

MODULE INFORMATION:

```

CODE AREA SIZE      = 005BH      91D
VARIABLE AREA SIZE = 0006H      6D
MAXIMUM STACK SIZE = 0004H      4D
93 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SCANNER

OBJECT MODULE PLACED IN SCAN

COMPILER INVOKED BY: PLM80 SCAN.PLM DEBUG PAGEWIDTH(130) OPTIMIZE OBJECT(SCAN)

```

1      $title ('Utility Command Line Scanner')
      scanner:
      do;

      $include(comlit.lit)

      =
2 1 = declare
      =      lit      literally      'literally',
      =      dcl      lit      'declare',
      =      true      lit      '0ffh',
      =      false     lit      '0',
      =      boolean   lit      'byte',
      =      forever   lit      'while true',
      =      cr        lit      '13',
      =      lf        lit      '10',
      =      tab        lit      '9',
      =      ctrlc     lit      '3',
      =      ff        lit      '12',
      =      page$len$offset lit      '1ch',
      =      nopage$mode$offset lit      '2Ch',
      =      sectorlen lit      '128';
      $include(mon.plm)
      =
      =      /* definitions for assembly interface module      */
3 1 = declare
      =      fcb (33) byte external,      /* default file control block */
      =      maxb address external,      /* top of memory */
      =      buff(128)byte external;      /* default buffer */
      =
4 1 = mon1: procedure(f,a) external;
5 2 = declare f byte, a address;
6 2 = end mon1;
      =
7 1 = mon2: procedure(f,a) byte external;
8 2 = declare f byte, a address;
9 2 = end mon2;
      =
10 1 = mon3: procedure(f,a) address external;
11 2 = declare f byte, a address;
12 2 = end mon3;
      =

13 1 dcl debug boolean initial (false);

14 1 dcl eob lit '0';      /* end of buffer */

      $include(fcb.lit)
      =
15 1 = declare
      =      f$drvusr      lit '0',      /* drive/user byte */
      =      f$name        lit '1',      /* file name */

```

CP/M plus v3.0
 COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # *CP3-135*

```

=      f$namelen      lit '8',      /* file name length      */
=      f$type         lit '9',      /* file type field       */
=      f$typelen      lit '3',      /* type length          */
=      f$rw           lit '9',      /* high bit is R/W attribute */
=      f$dirsyz       lit '10',     /* high bit is dir/sys attribute */
=      f$arc          lit '11',     /* high bit is archive attribute */
=      f$ex           lit '12',     /* extent               */
=      f$sl           lit '13',     /* module byte          */
=      f$rc           lit '15',     /* record count         */
=      f$diskmap      lit '16',     /* file disk map        */
=      diskmaplen     lit '16',     /* disk map length      */
=      f$drvusr2      lit '16',     /* fcb2                 */
=      f$name2        lit '17',
=      f$type2        lit '25',
=      f$rrrec        lit '33',     /* random record        */
=      f$rrreco       lit '35',     /* ' ' overflow        */
=

```

```
/* ----- Some routines used for diagnostics if debug mode is on ----- */
```

```

16 1  printchar: procedure(char) external;
17 2      declare char byte;
18 2  end printchar;

19 1  printb: procedure external;
20 2  end printb;

21 1  crlf: procedure external;
22 2  end crlf;

23 1  pdecimal: procedure(v,prec,zerosup) external;
      /* print value v, field size = (log10 prec) + 1 */
      /* with leading zero suppression if zerosup = true */
24 2      declare v address,      /* value to print      */
      prec address,             /* precision           */
      zerosup boolean,          /* zero suppression flag */
      d byte;                  /* current decimal digit */

25 2  end pdecimal;

```

```

/*
show$buf: procedure;
dcl i byte;
i = 1;
call crlf;
call moni(9,('buff = $'));
do while buff(i) <> 0;
    i = i + 1;
end;
buff(i) = '$';
call moni(9,buff(1));
buff(i) = 0;
end show$buf; */

```

```
/* ----- */
```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

26 1  white$space: procedure (str$adr) byte;
27 2      dcl str$adr address,
          str based str$adr (1) byte,
          i byte;
28 2      i = 0;
29 2      do while (str(i) = ' ') or (str(i) = tab);
30 3          i = i + 1;
31 3      end;
32 2      return(i);
33 2  end white$space;

34 1  delimiter: procedure(char) boolean;
35 2      dcl char byte;
36 2      if char = '[' or char = ']' or char = '(' or char = ')' or
          char = '=' or char = ',' or char = 0 then
37 2          return (true);
38 2      return(false);
39 2  end delimiter;

40 1  dcl string$marker lit '05ch';

41 1  debblank: procedure(buf$adr);
42 2      dcl (buf$adr,dest) address,
          buf based buf$adr (128) byte,
          (i,numspaces) byte,
          string boolean;

43 2      string = false;
44 2      if (numspaces := white$space(.buf(1))) > 0 then
45 2          call move(buf(0) - numspaces + 1,.buf(numspaces+1),.buf(1));
46 2      i = 1;
47 2      do while buf(i) <> 0;

/*      call show$buf;*/

48 3      do while ((numspaces := white$space(.buf(i))) = 0 and (buf(i) <> 0))
          and not string;
/*      call mon1(9,.(cr,lf,'2numspaces = $'));
      call pdecimal(numspaces,100,false);*/
/*      call show$buf;*/
49 4      if buf(i) = '' then
50 4          do;
51 5          string = true;
52 5          buf(i) = string$marker;
53 5          end;
54 4      i = i + 1;
55 4      end;

56 3      do while string and buf(i) <> 0;
57 4          if buf(i) = '' then
58 4              if buf(i+1) = '' then
59 4                  call move(buf(0) - i + 1,.buf(i+1),.buf(i));
          else
60 4              do;
61 5              buf(i) = string$marker;
62 5              string = false;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

PL/M-80 COMPILER UTILITY COMMAND LINE SCANNER

```

63 5          end;
64 4          i = i + 1;
65 4          end;

66 3          if (numspaces := white$space(.buf(i))) > 0 then
67 3              do;
/*              call moni(9,.(cr,lf,'numspaces = $'));
                call pdecimal(numspaces,100,false);*/
68 4              buf(i) = ' ';
69 4              dest = .buf(i+1);          /* save space for ', ' */
70 4              if i > 1 then
71 4                  if delimiter(buf(i-1)) or delimiter(buf(i+numspaces)) then
/* write over ' ' with */
72 4                      dest = dest - 1;    /* a = [ ] ( ) */

73 4              call move(((buf(0)+1)-(i+numspaces-1)),
                .buf(i+numspaces),dest);
74 4              if buf(i) = '' then
75 4                  string = true;
76 4              i = i + 1;
77 4          end;

78 3          end;
79 2          if buf(i - 1) = ' ' then      /* no trailing blanks */
80 2              buf(i - 1) = 0;
/* if debug then
            call show$buf; */
81 2          end deblank;

82 1          upper$case: procedure (buf$adr);
83 2              dcl buf$adr address,
                buf based buf$adr (1) byte,
                i byte;

84 2              i = 0;
85 2              do while buf(i) <> eob;
86 3                  if buf(i) >= 'a' and buf(i) <= 'z' then
87 3                      buf(i) = buf(i) - ('a' - 'A');
88 3                      i = i + 1;
89 3                  end;
90 2              end upper$case;

91 1          dcl option$max lit '11';
92 1          dcl done$scan lit '0ffffh';
93 1          dcl ident$max lit '11';
94 1          dcl token$max lit '11';

95 1          dcl t$null lit '0',
                t$param lit '1',
                t$option lit '2',
                t$modifier lit '4',
                t$identifier lit '8',
                t$string lit '16',
                t$numeric lit '32',
                t$filespec lit '64',
                t$error lit '128';

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

96 1    dcl pcb$base address;
97 1    dcl pcb based pcb$base structure (
        state address,
        scan$adr address,
        token$adr address,
        token$type byte,
        token$len byte,
        p$level byte,
        nxt$token byte);

98 1    dcl    scan$adr address,
        inbuf based scan$adr (1) byte,
        in$ptr byte,
        token$adr address,
        token based token$adr (1) byte,
        t$ptr byte,
        (char, nxtchar, tcount) byte;

99 1    digit: procedure (char) boolean;
100 2    dcl char byte;
101 2    return (char >= '0' and char <= '9');
102 2    end digit;

103 1    letter: procedure (char) boolean;
104 2    dcl char byte;
105 2    return (char >= 'A' and char <= 'Z');
106 2    end letter;

107 1    eat$char: procedure;
108 2    char = inbuf(in$ptr := inptr + 1);
109 2    nxtchar = inbuf(in$ptr + 1);
110 2    end eat$char;

111 1    put$char: procedure(charx);
112 2    dcl charx byte;
113 2    if pcb.token$adr <> 0ffffh then
114 2    token(t$ptr := t$ptr + 1) = charx;
115 2    end put$char;

116 1    get$identifier: procedure (max) byte;
117 2    dcl max byte;

118 2    tcount = 0;
    /* call mon1(9,.(cr,lf,'getindentifier$'));*/
119 2    if not letter(char) and char <> '$' then
120 2    return(tcount);
121 2    do while (letter(char) or digit(char) or char = '_' or
        char = '$') and tcount <= max;
122 3    call put$char(char);
123 3    call eat$char;
124 3    tcount = tcount + 1;
125 3    end;
126 2    do while letter(char) or digit(char) or char = '_'
        or char = '$';
127 3    call eat$char;
128 3    tcount = tcount + 1;
129 3    end;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 570
 PACIFIC GROVE, CA 93950

SER. # _____

```

130  2      pcb.token$type = t$identifier;
      /*      call mon1(9,.(cr,lf,'end of getident$')); */
131  2      pcb.token$len = tcount;
132  2      return(tcount);
133  2      end get$identifier;

134  1      file$char: procedure (x) boolean;
135  2          dcl x byte;
136  2          return(letter(x) or digit(x) or x = '*' or x = '?'
      or x = '_' or x = '$');
137  2      end file$char;

138  1      expand$wild$cards: procedure(field$size) boolean;
139  2          dcl (i,leftover,field$size) byte,
      save$inptr address;

140  2          field$size = field$size + t$ptr;
141  2      do while filechar(char) and t$ptr < field$size;
142  3          if char = '*' then
143  3              do; leftover = t$ptr;
144  4                  save$inptr = inptr;
145  4                  call eatchar;
146  4                  do while filechar(char);
147  5                      leftover = leftover + 1;
148  5                      call eatchar;
149  5                  end;
150  5                  if leftover >= field$size then /* too many chars */
151  4                      do; inptr = save$inptr;
152  4                          return(false);
153  5                      end;
154  5                      do i = 1 to field$size - leftover;
155  5                          call putchar('?');
156  5                      end;
157  4                      inptr = save$inptr;
158  4                  end;
159  4                  else
160  3                      call putchar(char);
161  3                      call eatchar;
162  3                  end;
163  2          return(true);
164  2      end expand$wild$cards;

166  1      get$file$spec: procedure boolean;
167  2          dcl i byte;
168  2          do i = 1 to f$name$len + f$type$len;
169  3              token(i) = ' ';
170  3          end;
171  2          if nextchar = ':' then
172  2              if char >= 'A' and char <= 'P' then
173  2                  do;
174  3                      call putchar(char - 'A' + 1);
175  3                      call eat$char; /* skip ':' */
176  3                      call eat$char; /* 1st char of file name */
177  3                  end;
178  2                  else
179  2                      return(false);
180  2                  else

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

179  2      call putchar(0);                /* use default drive */

180  2      if not (letter(char) or char = '$' or char = '_'
181          or char = '*' or char = '?' ) then /* no leading numerics */
182  2          if token(0) = 0 then          /* ambiguous with numeric token */
183          return(false);

183  2      if not expand$wild$cards(f$name$len) then
184  2          return(false);                /* blank name is illegal */
185  2      if char = '.' then
186  2          do; call eat$char;
188  3          if filechar(char) then
189  3              do; t$ptr = f$name$len;
191  4                  if not expand$wild$cards(f$type$len) then
192  4                      return(false);
193  4                  end;
194  3          end;

195  2      pcb.token$len = f$name$len + f$type$len + 1;
196  2      pcb.token$type = t$file$spec;
197  2      return(true);
198  2  end get$file$spec;

199  1  get$numeric: procedure(max) boolean;
200  2      dcl max byte;
201  2      if not digit(char) then
202  2          return(false);
203  2      do while digit(char) and pcb.token$len <= max and
204  3          char <> eob;
205  3          call putchar(char);
206  3          call eat$char;
207  3          pcb.token$len = pcb.token$len + 1;
208  2      end;
209  2      if char = 'H' or char = 'D' or char = 'B' then
210  2          if pcb.token$len < max then
211  3              do;
212  3                  call putchar(char);
213  3                  call eat$char;
214  3                  pcb.token$len = pcb.token$len + 1;
215  2              end;
216  2          else
217  2              return(false);
218  2          pcb.token$type = t$numeric;
219  2          return(true);
220  2  end get$numeric;

221  1  get$string: procedure(max) boolean;
222  2      dcl max byte;
223  2      if char <> string$marker then
224  2          return(false);
225  2      call eatchar;
226  2      do while char <> string$marker and char <> eob
227  3          and pcb.token$len < token$max;
228  3          call putchar(char);
229  3          call eatchar;
230  3          pcb.token$len = pcb.token$len + 1;
231  2      end;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

229 2      do while char <> string$marker and char <> eob;
230 3          call eat$char;
231 3      end;
232 2      if char <> string$marker then
233 2          return(false);
234 2      pcb.token$type = t$string;
235 2      call eat$char;
236 2      return(true);
237 2  end get$string;

238 1  get$token$all: procedure boolean;
239 2      dcl save$inptr byte;

/*      call mon1(9,.(cr,lf,'gettokenall$'));*/

240 2      save$inptr = in$ptr;
241 2      if get$file$spec then
242 2          return(true);

/*      call mon1(9,.(cr,lf,'gettokenall - no file$')); */
243 2      in$ptr = save$inptr - 1; /* need to re-scan, reset buffer pointers */
244 2      call eat$char;
245 2      t$ptr = 255;
246 2      call putchar(0);          /* zero drive byte */

247 2      if get$identifier(token$max) = 0 then
248 2          if not get$string(token$max) then
249 2              if not get$numeric(token$max) then
250 2                  return(false);
/*      call mon1(9,.(cr,lf,'end gettokenall$'));*/
251 2      return(true);
252 2  end get$token$all;

253 1  get$modifier: procedure boolean;
254 2      if char = ',' or char = ')' or char = 0 then
255 2          do;
256 3          pcb.token$type = t$modifier or t$null;
257 3          return(true);
258 3      end;
259 2      if get$token$all then
260 2          do;
261 3          pcb.token$type = pcb.token$type or t$modifier;
262 3          return(true);
263 3      end;
264 2      return(false);
265 2  end get$modifier;

266 1  get$option: procedure boolean;
267 2      call putchar(0);
268 2      if get$identifier(token$max) > 0 then
269 2          do;
270 3          pcb.token$type = pcb.token$type or t$option;
271 3          if pcb.token$len > token$max then
272 3              pcb.token$len = token$max;
273 3          return(true);
274 3      end;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

275 2      return(false);
276 2  end get$option;

277 1      get$param: procedure boolean;
278 2          if char = ',' or char = ')' or char = 0 then
279 2              do;
280 3                  pcb.token$type = t$param or t$null;
281 3                  return(true);
282 3              end;
283 2          if get$token$all then
284 2              do;
285 3                  pcb.token$type = pcb.token$type or t$param;
286 3                  return(true);
287 3              end;
288 2          return(false);
289 2  end get$param;

```

```

290 1      dcl gotatoken boolean;
291 1      dcl parens byte initial (0);

```

```

292 1      end$state: procedure boolean;
293 2          if gotatoken then
294 2              do;
295 3                  pcb.state = .end$state;
296 3                  return(true);
297 3              end;
298 2          pcb.token$type = t$null;
299 2          pcb.scan$adr = 0ffffh;
300 2          return(true);
301 2  end end$state;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

302 1      state8: procedure boolean reentrant;
303 2          if debug then do;
305 3              call mon1(9,.(cr,lf,'state8, char = $'));
306 3              call printchar(char); end;
308 2          if char = 0 then
309 2              return(end$state);
310 2          if char = ']' then
311 2              do;
312 3                  call eatchar;
313 3                  if char = ',' or nextchar = '(' or nextchar = ')' then
314 3                      return(state2);
315 3                  else if char = 0 then
316 3                      return(end$state);
317 3                  else
318 3                      return(state1);
319 2              end;
319 2          else if char = ' ' or char = ',' then
320 2              do;
321 3                  call eatchar;
322 3                  return(state3);
323 3              end;
324 2          return(state3);
325 2  end state8;

326 1      state7: procedure boolean reentrant;
327 2          if debug then do;

```

```

329 3      call mon1(9,.(cr,lf,'state7, char = $'));
330 3      call printchar(char); end;
332 2      if char = 0 then
333 2          return(end$state);
334 2      if char = ' ' or char = ',' then
335 2      do;
336 3          call eat$char;
337 3          return(state6);
338 3      end;
339 2      else
340 2          if char = ')' then
341 3              call eat$char;
342 3              return(state8);
343 3          end;
344 2      return(false);
345 2  end state7;

346 1  state6: procedure boolean reentrant;
347 2      if debug then do;
348 3          call mon1(9,.(cr,lf,'state6, char = $'));
349 3          call printchar(char); end;
350 3      if gotatoken then
351 2      do;
352 3          pcb.state = .state6;
353 3          pcb.nxt$token = t$modifier;
354 3          return(true);
355 3      end;
356 2      if (gotatoken := get$modifier) then
357 2          return(state7);
358 2      return(false);
359 2  end state6;

360 1  state5: procedure boolean reentrant;
361 2      if debug then do;
362 3          call mon1(9,.(cr,lf,'state5, nextchar = $'));
363 3          call printchar(nextchar); end;
364 2      if char = '(' then
365 2      do;
366 3          call eat$char;
367 3          return(state6);
368 3      end;
369 2      if gotatoken then
370 2      do;
371 3          pcb.state = .state5;
372 3          pcb.nxt$token = t$modifier;
373 3          return(true);
374 3      end;
375 2      if (gotatoken := get$modifier) then
376 2          return(state8);
377 2      return(false);
378 2  end state5;

379 1  state4: procedure boolean reentrant;
380 2      dcl temp byte;
381 2      if debug then do;
382 3          call mon1(9,.(cr,lf,'state4, char = $'));

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

388 3      call printchar(char); end;
390 2      if char = 0 then
391 2          return(end$state);
392 2      temp = char;
393 2      call eatchar;
394 2      if temp = ',' or temp = ' ' then
395 2          return(state3);
396 2      if temp = ']' then
397 2          if char = '(' or char = ',' or char = ')' then
398 2              return(state2);
399 2          else if char = 0 then
400 2              return(end$state);
           else
401 2              return(state1);
402 2      if temp = '=' then
403 2          return(state5);
404 2      return(false);
405 2  end state4;

406 1  state3: procedure boolean reentrant;
407 2      if debug then do;
409 3      call mon1(9,.(cr,lf,'state3, char = $'));
410 3      call printchar(char); end;
412 2      if gotatoken then
413 2          do;
414 3          pcb.state = .state3;
415 3          pcb.nxt$token = t$option;
416 3          return(true);
417 3      end;
418 2      if (pcb.plevel := parens) > 128 then
419 2          return(false);
420 2      if (gotatoken := get$option) then
421 2          return(state4);
422 2      return(false);
423 2  end state3;

424 1  state2: procedure boolean reentrant;
425 2      if debug then do;
427 3      call mon1(9,.(cr,lf,'state2, char = $'));
428 3      call printchar(char); end;
430 2      do while char = ')' or char = 0;
431 3          if char = 0 then
432 3              return(end$state);
433 3          call eat$char;
434 3          parens = parens - 1;
435 3      end;
436 2      if char = '[' then
437 2          do;
438 3          call eat$char;
439 3          return(state3);
440 3      end;
441 2      if char = ' ' or char = ',' or char = '(' then
442 2          do;
443 3          if char = '(' then
444 3              parens = parens + 1;
445 3          call eat$char;
446 3          return(state1);

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

447 3      end;
448 2      return(state1);
449 2      end state2;

450 1      state1: procedure boolean reentrant;
451 2          if debug then do;
453 3              call moni(9,.(cr,lf,'state1, char = $'));
454 3              call printchar(char); end;

456 2      if gotatoken then
457 2      do;
458 3          pcb.nxt$token = t$param;
459 3          pcb.state = .state1;
460 3          return(true);
461 3      end;
462 2      do while char = '(' ;
463 3          parens = parens + 1;
464 3          call eat$char;
465 3      end;
466 2      if (pcb.plevel := parens) > 128 then
467 2          return(false);
468 2      if (gotatoken := get$param) then
469 2          return(state2);
470 2      return(false);
471 2      end state1;

472 1      start$state: procedure boolean;
473 2      if char = '@' then do;
475 3          debug = true;
476 3          call eat$char;
477 3          call moni(9,.(cr,lf,'startstate, char = $'));
478 3          call printchar(char); end;

480 2      if char = 0 then
481 2          return(end$state);
482 2      if char = ')' then
483 2          return(false);
484 2      if char = '(' then
485 2      do;
486 3          parens = parens + 1;
487 3          call eat$char;
488 3          return(state1);
489 3      end;
490 2      if char = '[' then
491 2      do;
492 3          call eat$char;
493 3          return(state3);
494 3      end;
495 2      if (gotatoken := get$param) then
496 2          return(state2);
497 2      return(false);
498 2      end start$state;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* display\$all: procedure; /* called if debug set */

```

/* call moni(9,.(cr,lf,'scanadr=$'));
call pdecimal(pcb.scanadr,10000,false);

```

```

call mon1(9,(',', tadr='$'));
call pdecimal(pcb.token$adr,10000, false);
call mon1(9,(',', tlen='$'));
call pdecimal(double(pcb.token$len),100, false);
call mon1(9,(',', ttype='$'));
call pdecimal(double(pcb.token$type),100,false);
call mon1(9,(',', plevel='$'));
call pdecimal(double(pcb.plevel),100,false);
call mon1(9,(',', ntok='$'));
call pdecimal(double(pcb.nxt$token),100,false);

if (pcb.token$type and t$option) <> 0 then
    call mon1(9,((cr,lf,'option  =')));
if (pcb.token$type and t$param) <> 0 then
    call mon1(9,((cr,lf,'parm   =')));
if (pcb.token$type and t$modifier) <> 0 then
    call mon1(9,((cr,lf,'modifier=')));

if (pcb.token$type and t$filespec) <> 0 then
do;
    if fcb(0) = 0 then
        call print$char('0');
    else call print$char(fcb(0) + 'A' - 1);
    call print$char(':');
    fcb(12) = '$';
    call mon1(9,fcb(1));
    call mon1(9,((      (filespec)$'));
end;
if ((pcb.token$type and t$string) or (pcb.token$type and
    t$identifier) or (pcb.token$type and t$numeric)) <> 0 then
do;
    fcb(pcb.token$len + 1) = '$';
    call mon1(9,fcb(1));
end;
if pcb.token$type = t$error then
do;
    call mon1(9,((cr,lf,'scanner error$'));
    return;
end;

if (pcb.token$type and t$identifier) <> 0 then
    call mon1(9,((      (identifier)$'));
if (pcb.token$type and t$string) <> 0 then
    call mon1(9,((      (string)$'));
if (pcb.token$type and t$numeric) <> 0 then
    call mon1(9,((      (numeric)$'));

if (pcb.nxt$token and t$option) <> 0 then
    call mon1(9,((cr,lf,'nxt tok = option  $'));
if (pcb.nxt$token and t$param) <> 0 then
    call mon1(9,((cr,lf,'nxt tok = parm   $'));
if (pcb.nxt$token and t$modifier) <> 0 then
    call mon1(9,((cr,lf,'nxt tok = modifier$'));
call crlf;

```

```
end display$all; %/
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93050

SER. # _____

```

499 1      scan: procedure (pcb$adr) public;

500 2          dcl status boolean,
              pcb$adr address;

501 2          pcb$base = pcb$adr;
502 2          scan$adr = pcb.scan$adr;
503 2          token$adr = pcb.token$adr;

504 2          in$ptr, t$ptr = 255;
505 2          call eatchar;

506 2          gotoken = false;
507 2          pcb.next$token = t$null;
508 2          pcb.token$len = 0;

509 2          if pcb.token$type = t$error then      /* after one error, return */
510 2              return;                          /* on any following calls */
511 2          else if pcb.state = .start$state then
512 2              status = start$state;
513 2          else if pcb.state = .state#1 then
514 2              status = state#1;
515 2          else if pcb.state = .state#3 then
516 2              status = state#3;
517 2          else if pcb.state = .state#5 then
518 2              status = state#5;
519 2          else if pcb.state = .state#6 then
520 2              status = state#6;
521 2          else if pcb.state = .end$state then    /* repeated calls go here */
522 2              status = end$state;                /* after first end$state */
523 2          else
524 2              status = false;
525 2          if not status then
526 2              pcb.token$type = t$error;

526 2          if pcb.scan$adr < 0ffffh then
527 2              pcb.scan$adr = pcb.scan$adr + inptr;
              /* if debug then
              call display$all; */

528 2      end scan;

529 1      scan$init: procedure(pcb$adr) public;
530 2          dcl pcb$adr address;

531 2          pcb$base = pcb$adr;
532 2          call deblank(pcb.scan$adr);
533 2          call upper$case(pcb.scan$adr := pcb.scan$adr + 1);
534 2          pcb.state = .start$state;
535 2      end scan$init;

536 1      end scanner;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

PL/M-80 COMPILER UTILITY COMMAND LINE SCANNER

CODE AREA SIZE = 0E8DH 3725D
VARIABLE AREA SIZE = 0030H 48D
MAXIMUM STACK SIZE = 001FH 31D
787 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SEARCH

OBJECT MODULE PLACED IN SEARCH

COMPILER INVOKED BY: PLM80 SEARCH,PLM DEBUG PAGEWIDTH(130) OPTIMIZE OBJECT(SEARCH)

```

1      $title ('SDIR - Search For Files')
      search;
      do;

          /* search module for extended dir */

          $include (comlit.lit)

          =
2      1 = declare
          =         lit           literally      'literally',
          =         dcl           lit           'declare',
          =         true          lit           '0ffh',
          =         false         lit           '0',
          =         boolean       lit           'byte',
          =         forever       lit           'while true',
          =         cr            lit           '13',
          =         lf            lit           '10',
          =         tab           lit           '9',
          =         ctrlc         lit           '3',
          =         ff            lit           '12',
          =         page$len$offset lit         '1ch',
          =         nopage$mode$offset lit       '2Ch',
          =         sectorlen     lit           '128';
          $include (mon.plm)

          =
          =           /* definitions for assembly interface module      */

3      1 = declare
          =         fcb (33) byte external,      /* default file control block */
          =         maxb address external,        /* top of memory              */
          =         buff(128)byte external;      /* default buffer             */
          =
4      1 = mon1: procedure(f,a) external;
5      2 =         declare f byte, a address;
6      2 =         end mon1;
          =
7      1 = mon2: procedure(f,a) byte external;
8      2 =         declare f byte, a address;
9      2 =         end mon2;
          =
10     1 = mon3: procedure(f,a) address external;
11     2 =         declare f byte, a address;
12     2 =         end mon3;
          =

13     1      dcl debug boolean external;

14     1      dcl first$pass boolean external;
15     1      dcl get$all$dir$entries boolean external;
16     1      dcl usr$vector address external;
17     1      dcl active$usr$vector address external;
18     1      dcl used$de address public;          /* used directory entries    */
19     1      dcl filesfound address public;        /* num files collected in memory */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SFR. #

```

        $include(fcb.lit)
=
20 1 = declare
=     f$drvusr      lit '0',      /* drive/user byte      */
=     f$name        lit '1',      /* file name            */
=     f$namelen     lit '8',      /* file name length     */
=     f$type        lit '9',      /* file type field      */
=     f$typelen     lit '3',      /* type length          */
=     f$rw          lit '9',      /* high bit is R/W attribute */
=     f$dirsyz      lit '10',     /* high bit is dir/sys attribute */
=     f$arc         lit '11',     /* high bit is archive attribute */
=     f$ex          lit '12',     /* extent               */
=     f$sl          lit '13',     /* module byte          */
=     f$rc          lit '15',     /* record count         */
=     f$diskmap     lit '16',     /* file disk map        */
=     diskmaplen    lit '16',     /* disk map length      */
=     f$drvusr2     lit '16',     /* fcb2                 */
=     f$name2       lit '17',
=     f$type2       lit '25',
=     f$rrc         lit '33',     /* random record        */
=     f$rrco        lit '35';    /* ' ' overflow        */
=
        $include(xfcb.lit)
=
21 1 = declare /* XFCB */
=     xfcb$type     lit '10h',    /* identifier on disk   */
=     xf$passmode   lit '12',    /* pass word protection mode */
=     xf$pass       lit '16',    /* XFCB password       */
=     passlen       lit '8',     /* password length      */
=     xf$create     lit '24',    /* creation/access time stamp */
=     xf$update     lit '28';    /* update time stamp    */
=
22 1 = declare /* directory label: special case of XFCB */
=     dl$labeltype  lit '20h',    /* identifier on disk   */
=     dl$password   lit '128',    /* masks on data byte   */
=     dl$access     lit '64',
=     dl$update     lit '32',
=     dl$makexfcb   lit '16',
=     dl$exists     lit '1';
=
23 1 = declare /* password mode of xfcb */
=     pm$read       lit '80h',
=     pm$write      lit '40h',
=     pm$delete     lit '20h';
=
24 1 = declare
=     sfcb$type     lit '21h',
=     deleted$type  lit '0E5H';
=
        $include (search.lit)
=
25 1 = declare /* what kind of file user wants to find */
=     find$structure lit 'structure (
=     dir byte,
=     sys byte,

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

      =      ro byte,
      =      rw byte,
      =      pass byte,
      =      xfc byte,
      =      nonxfc byte,
      =      exclude byte)';
      =
26  1  =  declare
      =      max$search$files literally '10';
      =
27  1  =  declare
      =      search$structure lit 'structure(
      =      drv byte,
      =      name(8) byte,
      =      type(3) byte,
      =      anyfile boolean)';      /* match on any drive if true */
      =
28  1  dcl find find$structure external;      /* what kind of files to look for */
29  1  dcl num$search$files byte external;
30  1  dcl search (max$search$files) search$structure external;
      /* file specs to match on */

      /* other globals */

31  1  dcl cur$usr byte external,
      cur$drv byte external,      /* current drive */
      dir$label byte public;      /* directory label for RDOS 3.0 */

      /* ----- RDOS calls ----- */

32  1  read$char: procedure byte;
33  2      return mon2 (1,0);
34  2  end read$char;

      /* ----- in sort.plm ----- */

35  1  mult23: procedure(f$info$index) address external;
36  2      dcl f$info$index address;
37  2  end mult23;

      /* ----- in util.plm ----- */

38  1  print: procedure(string$adr) external;
39  2      dcl string$adr address;
40  2  end print;

41  1  print$char: procedure(char) external;
42  2      dcl char byte;
43  2  end print$char;

44  1  pdecimal: procedure(val, prec, zsup) external;
45  2      dcl (val, prec) address;
46  2      dcl zsup boolean;
47  2  end pdecimal;

```

CP/m plus Ver 3.0

COPYRIGHT © 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # *CP3-135*

```

48 1    printfn: procedure(fnameadr) external;
49 2        dcl fnameadr address;
50 2    end printfn;

51 1    crlf: procedure external; /* print carriage return, linefeed */
52 2    end crlf;

53 1    add3byte: procedure(byte3adr,num) external;
54 2        dcl (byte3adr,num) address;
55 2    end add3byte;

        /* add three byte number to 3 byte accumulator */
56 1    add3byte3: procedure(totalb,numb) external;
57 2        dcl (totalb,numb) address;
58 2    end add3byte3;

        /* divide 3 byte value by 8 */
59 1    shr3byte: procedure(byte3adr) external;
60 2        dcl byte3adr address;
61 2    end shr3byte;

        /* ----- In dpb86.plm ----- */

#include(dpb.lit)
=
= /* indices into disk parameter block, used as parameters to dpb procedure */
=
62 1 = dcl      spt$w      lit      '0',
=          blkshf$b     lit      '2',
=          blkmsk$b     lit      '3',
=          extmsk$b     lit      '4',
=          blkmax$w     lit      '5',
=          dirmax$w     lit      '7',
=          dirblk$w     lit      '9',
=          chksiz      lit      '11',
=          offset$w     lit      '13';
=

63 1    dcl k$per$block byte external; /* set in dpb module */

64 1    base$dpb: procedure external;
65 2    end base$dpb;

66 1    dpb$byte: procedure(param) byte external;
67 2        dcl param byte;
68 2    end dpb$byte;

69 1    dpb$word: procedure(param) address external;
70 2        dcl param byte;
71 2    end dpb$word;

        /* ----- Some Utility Routines ----- */

72 1    check$console$status: procedure byte;
73 2        return mon2 (11,0);

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

74 2   end check$console$status;

75 1   search$first: procedure (fcb$address) byte public;
76 2       declare fcb$address address;           /* shared with disp.plm */
77 2       return mon2 (17,fcb$address);           /* for short display */
78 2   end search$first;

79 1   search$next: procedure byte public;           /* shared with disp.plm */
80 2       return mon2 (18,0);
81 2   end search$next;

82 1   terminate: procedure external;               /* in main.plm */
83 2   end terminate;

84 1   set$vec: procedure(vector,value) external;    /* in main.plm */
85 2   dcl vector address,
      value byte;
86 2   end set$vec;

87 1   break: procedure public;                     /* shared with disp.plm */
88 2       dcl x byte;
89 2       if check$console$status then
90 2           do;
91 3               x = read$char;
92 3               call terminate;
93 3           end;
94 2   end break;

      /* ----- file information record declaration ----- */

      $include(finfo.lit)

      =
      = /* file info record for SDIR - note if this structure changes in size */
      = /* the multXX: routine in the sort.plm module must also change */
      =

95 1 = declare
      =         f$info$structure lit 'structure(
      =             usr byte, name (8) byte, type (3) byte, onekblocks address,
      =             kbytes address, recs$lword address, recs$hbyte byte,
      =             hash$link address, x$i$adr address)';

96 1 = declare
      =         x$info$structure lit 'structure (
      =             create (4) byte,
      =             update (4) byte,
      =             passmode byte)';
      =

97 1   declare
      buf$fcb$adr address public,           /* index into directory buffer */
      buf$fcb based buf$fcb$adr (32) byte,
                                         /* fcb template for dir */
      (first$f$i$adr, f$i$adr, last$f$i$adr) address public,
                                         /* indices into file$info array */
      file$info based f$i$adr f$info$structure,
      sfc$b$adr address,
      dir$type based sfc$b$adr byte,

```

COPYRIGHT ©1982
 . DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SFR. #

```

sfcbs$present byte public,
x$i$adr address public,
x$cb$info based x$i$adr x$info$structure;

98 1  compare: procedure(length, str1$adr, str2$adr) boolean;
99 2      dcl (length,i) byte,
        (str1$adr, str2$adr) address,
        str1 based str1$adr (1) byte,
        str2 based str2$adr (1) byte;
        /* str2 is the possibly wildcarded filename we are looking for */

100 2      do i = 0 to length - 1;
101 3          if ((str1(i) and 7fh) <> (str2(i) and 7fh)) and str2(i) <> '?' then
102 3              return(false);
103 3      end;
104 2      return(true);
105 2  end compare;

106 1  match: procedure boolean public;
107 2      dcl i byte,
        temp address;
108 2      if (i := (buf$fcb(f$drvusr) and 0fh)) <> cur$usr then
109 2          if not get$all$dir$entries then /* Not looking for this user */
110 2              return(false); /* and not buffering all other*/
        else /* specified user files on */
111 2          do; temp = 0; /* this drive. */
113 3              call set$vec(.temp,i);
114 3              if (temp and usr$vector) = 0 then /* Getting all dir entries, */
115 3                  return(false); /* with user number corresp'g */
116 3          end; /* to a bit on in usr$vector */

117 2      if usr$vector <> 0 and i <> 0 and first$pass <> 0 then
118 2          call set$vec(.active$usr$vector,i); /* skip cur$usr files */
        /* build active usr vector for this drive */

119 2      do i = 0 to num$search$files - 1;
120 3          if search(i).drv = 0fh or search(i).drv = cur$drv then
        /* match on any drive if 0fh */
121 3              if search(i).anyfile = true then
122 3                  return(not find.exclude); /* file found */
123 3              else if compare(11,buf$fcb(f$name),search(i).name(0)) then
124 3                  return(not find.exclude); /* file found */
        end;
126 2      return(find.exclude); /* file not found */
127 2  end match; /* find.exclude = the exclude option value */

128 1  dcl hash$table$size lit '128', /* must be power of 2 */
        hash$table (hash$table$size) address at (.memory), /* must be initialized on each*/
        hash$entry$adr address, /* disk scan */
        hash$entry based hash$entry$adr address; /* where to put a new entry's */
        /* address */

129 1  hash$look$up: procedure boolean;
130 2      dcl (i,found,hash$index) byte;
131 2      hash$index = 0;
132 2      do i = f$name to f$namelen + f$typelen;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

133 3      hash$index = hash$index + (buf$fcbl(i) and 7fh); /* attributes may */
134 3      end; /* only be set w/ 1st extent */
135 2      hash$index = hash$index + cur$usr;
136 2      hash$index = hash$index and (hash$tblsize - 1);
137 2      hash$entry$adr = .hash$tbl[hash$index]; /* put new entry in table if */
138 2      f$i$adr = hash$tbl[hash$index]; /* unused ( = 0) */

139 2      found = false;
140 2      do while f$i$adr < 0 and not found;
141 3          if file$info.usr = (buf$fcbl(f$drvusr) and 0fh) and
              compare(f$namelen + f$typelen, file$info.name(0), buf$fcbl(f$name))
              then
142 3              found = true;
              else
143 3                  /* table entry used - collison */
144 4                  do; hash$entry$adr = .file$info.hash$link; /* resolve by linked */
145 4                  f$i$adr = file$info.hash$link; /* list */
146 4                  end;
147 3      end;
148 2      if f$i$adr = 0 then
149 2          return(false); /* didn't find it, used hash$entry to keep new info */
150 2      else return(true); /* found it, file$info at matched entry */
151 2      end hash$look$up;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

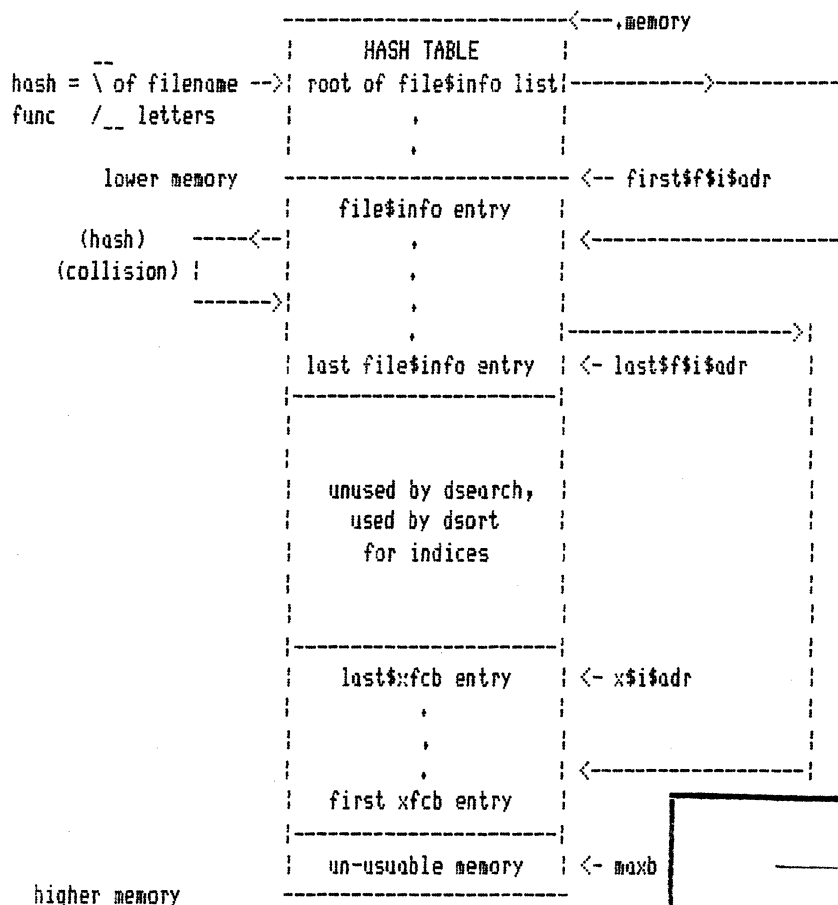
PACIFIC GROVE, CA 93950

SER. # _____

```

$eject
152 1 store$file$info: procedure boolean;
    /* Look for file name of last found fcb or xfc in fileinfo */
    /* array, if not found put name in fileinfo array. Copy other */
    /* info to fileinfo or xfcinfo. The lookup is hash coded with */
    /* collisions handled by linking up file$info records through */
    /* the hash$link field of the previous file$info record. */
    /* The file$info array grows upward in memory and the xfcinfo */
    /* grows downward. */
    /*

```



```

153 2 dcl (i, j, d$map$cnt) byte,
      temp address;

154 2 store$file: procedure;
155 3 call move(f$namelen + f$typelen, .buf$fcb(f$name), file$info.name);
      /* attributes are not in XFCBs to copy again in case */
      /* XFCB came first in directory */

156 3 file$info.name(f$arc-1) = file$info.name(f$arc-1) and buf$fcb(f$arc);
      /* 0 archive bit if it is 0 in any dir entry */
157 3 d$map$cnt = 0; /* count kilobytes for current dir entry */
158 3 i = 1; /* 1 or 2 byte block numbers ? */
159 3 if ddb$word(blk$max$w) > 255 then
160 3 i = 2;
161 3 do j = f$diskmap to f$diskmap + diskmaplen - 1 by i;

```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. 11


```

162 4      temp = buf$fcbl(j);
163 4      if i = 2 then                /* word block numbers */
164 4          temp = temp or buf$fcbl(j+1);
165 4      if temp < 0 then              /* allocated */
166 4          d$map$cnt = d$map$cnt + 1;
167 4      end;
168 3      if d$map$cnt > 0 then
169 3      do;
170 4          call add3byte
              (.file$info.recs$lword,
              d$map$cnt * (dpb$byte(blkmsk$b) + 1) -
              ( (128 - buf$fcbl(f$rc)) and dpb$byte(blkmsk$b) ) );
171 4      file$info.onekblocks = file$info.onekblocks +
              d$map$cnt * k$per$block -
              shr( (128 - buf$fcbl(f$rc)) and dpb$byte(blkmsk$b), 3 );
              /* treat each directory entry separately for sparse files */
              /* if copied to single density diskette, the number of 1kblocks */
172 4      file$info.kbytes = file$info.kbytes + d$map$cnt * k$per$block;
173 4      end;
174 3      end;

175 2      if buf$fcbl(f$drvusr) <> sfcbl$type then do; /* don't put SFCB's in table */
177 3          if not hash$look$up then                /* not in table already */
              /* hash$entry is where to put adr of new entry */
178 3          do;                                     /* copy to new position in file info array */
179 4              if (temp := mult23(files$found + 1)) > x$i$adr then
180 4                  return(false);                /* out of memory */
181 4              if (temp < first$f$i$adr) then
182 4                  return(false);                /* wrap around - out of memory */
183 4              f$i$adr = (last$f$i$adr := last$f$i$adr + size(file$info));
184 4              files$found = files$found + 1;
185 4              call move(f$namelen + f$type$len, .buf$fcbl(f$name), .file$info.name);
186 4              file$info.usr = buf$fcbl(f$drvusr) and 0fh;
187 4              file$info.onekblocks, file$info.kbytes, file$info.recs$lword,
                  file$info.recs$hbyte, file$info.x$i$adr, file$info.hash$link = 0;
188 4              hash$entry = f$i$adr;                /* save the address of file$info */
189 4          end;                                     /* zero totals for the new file */
190 3      end;

              /* else hash$lookup has set f$i$adr to the file entry already in the */
              /* hash table */
              /* save sfcbl, xfcbl or fcb type info */

191 2      if sfcbl$present then do;
193 3          if (buf$fcbl(f$drvusr) and xfcbl$type) = 0 then do;
195 4              if buf$fcbl(f$drvusr) <> sfcbl$type then do;
              /* store sfcbl info into xfcbl table */
197 5              if buf$fcbl(f$ex) <= dpb$byte(extmsk$b) then do;
199 6                  if last$f$i$adr + size(file$info) > x$i$adr - size(xfcbl$info) then
200 6                      return(false);            /* out of memory */
201 6                  x$i$adr = x$i$adr - size(xfcbl$info);
202 6                  call move(9, sfcbl$adr, xfcbl$info.create);
203 6                  file$info.x$i$adr = x$i$adr;
204 6              end; /* extent check */
205 5              call store$file;
206 5          end;
207 4      end;

```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

208 3      end;
209 2      else do;     /* no SFCB's present */
210 3          if (buf$fcbl(f$drvusr) and xfcbl$type) <> 0 then
211 3              do;     /* XFCB     */
/*
            if last$f$adr + size(file$info) > x$adr - size(xfcbl$info) then
                return(false);
            x$adr = x$adr - size(xfcbl$info);
            call move(8, buf$fcbl(xf$create), xfcbl$info.create);
            xfcbl$passmode = buf$fcbl(xf$passmode);
            file$info.x$adr = x$adr;
        */
212 4      end;
213 3      else do;
214 4          call store$file;     /* must be a regular fcb then */
215 4      end;
216 3      end;
217 2      return(true);     /* success     */
218 2      end store$file$info;

/* Module Entry Point */

219 1      get$files: procedure public;     /* with one scan through directory get     */
220 2          dcl dcnt byte;     /* files from currently selected drive     */

221 2          call print(, (cr, lf, 'Scanning Directory...', cr, lf, '$'));
222 2          last$f$adr = first$f$adr - size(file$info);
/* after hash table     */
/* last$f$adr is the address of the highest file info record     */
/* in memory     */

223 2          do dcnt = 0 to hash$tblsize - 1;     /* init hash table     */
224 3              hash$tbl(dcnt) = 0;
225 3          end;

226 2          x$adr = maxb;     /* top of mem, put xfcbl info here     */
227 2          call base$dcb;
228 2          dir$label, filesfound, used$de = 0;

229 2          fcb(f$drvusr) = '?';     /* match all dir entries     */
230 2          dcnt = search$first(fcb);
231 2          sfcbl$adr = 96 + .buff;     /* determine if SFCB's are present */

232 2          if dir$type = sfcbl$type then
233 2              sfcbl$present = true;
          else
234 2              sfcbl$present = false;

235 2          do while dcnt <> 255;
236 3              buf$fcbl$adr = shl(dcnt and 11b, 5) + .buff;     /* dcnt mod 4 * 32
SER. #

237 3              if sfcbl$present then
238 3                  sfcbl$adr = 97 + (dcnt * 10) + .buff;     /* SFCB time & date stamp adr */

239 3              if buf$fcbl(f$drvusr) <> deleted$type then
240 3                  do;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

PL/M-80 COMPILER SDIR - SEARCH FOR FILES

```

241 4      used$de = used$de + 1;

242 4      if buf$fcb(f$drvusr) = dirlabel$type then /* dir label ? */
243 4      dir$label = buf$fcb(f$ex); /* save label info */
      else
244 4      if (match) then
245 4      do;
246 5      if not store$file$info then /* store fcb or xfc info */
247 5      do; /* out of space */
248 6      call print (('Out of Memory',cr,lf,$'));
249 6      return;
250 6      end; /* not store$file$info */

251 5      end; /* else if match */

      end; /* buf$fcb(f$drvusr) <> deleted$type */

253 3      call break;
254 3      dcnt = search$next; /* to next entry in directory */

255 3      end; /* of do while dcnt <> 255 */
256 2      end get$files;

257 1      search$init: procedure public; /* called once from main.plm */

258 2      if (first$f$i$adr := (.hash$table + size(hash$table))) + size(file$info)
      > maxb then
259 2      do;
260 3      call print (('Not Enough Memory',cr,lf,$'));
261 3      call terminate;
262 3      end;
263 2      end search$init;

264 1      end search;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0653H  1619D
VARIABLE AREA SIZE  = 0029H   41D
MAXIMUM STACK SIZE  = 000CH   12D
564 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982 DIGITAL RESEARCH P. O. BOX 579 PACIFIC GROVE, CA 93950 SER. # _____
--

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SORT

OBJECT MODULE PLACED IN SORT

COMPILER INVOKED BY: PLM80 SORT.PL M DEBUG PAGEWIDTH(130) OPTIMIZE OBJECT(SORT)

```

1      $title ('SDIR - Sort Module')
      sort:
      do;

          /* sort module for extended dir */

          $include(comlit.lit)

          =
2      1 = declare
          =      lit      literally      'literally',
          =      dcl      lit      'declare',
          =      true     lit      '0ffh',
          =      false    lit      '0',
          =      boolean   lit      'byte',
          =      forever   lit      'while true',
          =      cr        lit      '13',
          =      lf        lit      '10',
          =      tab       lit      '9',
          =      ctrlc     lit      '3',
          =      ff        lit      '12',
          =      page$len$offset lit '1ch',
          =      nopage$mode$offset lit '2Ch',
          =      sectorlen lit      '128';

3      1      print: procedure(str$adr) external; /* in util.plm */
4      2      dcl str$adr address;
5      2      end print;

6      1      dcl sorted boolean public;          /* set by this module if successful sort */

          $include(finfo.lit)

          =
          =      /* file info record for SDIR - note if this structure changes in size */
          =      /* the multXX: routine in the sort.plm module must also change */
          =

7      1 = declare
          =      f$info$structure lit 'structure(
          =          usr byte, name (8) byte, type (3) byte, onekblocks address,
          =          kbytes address, recs$lword address, recs$hbyte byte,
          =          hash$link address, x$i$adr address)';

8      1 = declare
          =      x$info$structure lit 'structure (
          =          create (4) byte,
          =          update (4) byte,
          =          passmode byte)';
          =

9      1      declare
          =      buf$fc$adr address external, /* index into directory buffer */
          =      buf$fc$based buf$fc$adr (32) byte,
          =          /* fcb template for dir */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

(f%i$adr, first$f%i$adr, last$f%i$adr, x%i$adr, filesfound)
    address external,
                                /* indices into file$info array */
    file$info based f%i$adr f$info$structure,

    mid$adr address,
    mid$file$info based mid$adr f$info$structure;

10 1    mult23: procedure(index) address public;
11 2        dcl index address; /* return address of file$info numbered by index */
12 2        return shl(index, 4) + shl(index,2) + shl(index,1) + index + first$f%i$adr;
        /* index * size(file$info) + base of file$info array */
13 2    end mult23;

14 1    lessthan: procedure( str1$adr, str2$adr) boolean;
15 2        dcl (i,c1,c2) byte, /* true if str1 < str2 */
        (str1$adr, str2$adr) address, /* sorting on name and type field */
        str1 based str1$adr (1) byte, /* only, assumed to be first in */
        str2 based str2$adr (1) byte; /* file$info record */
16 2        do i = 1 to 11;
17 3            if (c1:=(str1(i) and 7fh) <> (c2:=(str2(i) and 7fh)) then
18 3                return(c1 < c2);
19 3        end;
20 2        return(false);
21 2    end lessthan;

22 1    dcl f%i$indices$base address public,
        f%i$indices based f%i$indices$base (1) address;

23 1    qsort: procedure(l,r); /* no recursive quick sort, sorting largest */
24 2        dcl (l,r,i,j,temp) address, /* partition first */
        stack$size lit '14', /* should always be able to sort 2 ** stack$size */
        stack (stack$size) structure (l address, r address),
        sp byte;

25 2        sp = 0; stack(0).l = l; stack(0).r = r;

28 2        do while sp < stack$size - 1;
29 3            l = stack(sp).l; r = stack(sp).r; sp = sp - 1;
32 3            do while l < r;
33 4                i = l; j = r;
35 4                mid$adr = mult23(f%i$indices(shr(l+r,1)));
36 4                do while i <= j;
37 5                    f%i$adr = mult23(f%i$indices(i));
38 5                    do while lessthan(f%i$adr,mid$adr);
39 6                        i = i + 1;
40 6                        f%i$adr = mult23(f%i$indices(i));
41 6                    end;
42 5                    f%i$adr = mult23(f%i$indices(j));
43 5                    do while lessthan(mid$adr,f%i$adr);
44 6                        j = j - 1;
45 6                        f%i$adr = mult23(f%i$indices(j));
46 6                    end;
47 5                    if i <= j then
48 5                        do; temp = f%i$indices(i); f%i$indices(i) = f%i$indices(j);
51 6                            f%i$indices(j) = temp;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

52 6          i = i + 1;
53 6          if j > 0 then j = j - 1;
55 6          end;
56 5          end; /* while i <= j */
57 4          if j - 1 < r - i then /* which partition is larger */
58 4          do; if i < r then
60 5              do; sp = sp + 1; stack(sp).l = i; stack(sp).r = r;
64 6              end;
65 5              r = j; /* continue sorting left partition */
66 5          end;
          else
67 4          do; if l < j then
69 5              do; sp = sp + 1; stack(sp).l = l; stack(sp).r = j;
73 6              end;
74 5              l = i; /* continue sorting right partition */
75 5          end;
76 4          end; /* while l < r */
77 3          end; /* while sp < stack$size - 1 */
78 2          if sp > 255 then
79 2              call print(, (cr, lf, lf, 'Sort Stack Overflow', cr, lf, '$'));
80 2          else sorted = true;
81 2          end qsort;

82 1          sort: procedure public;
83 2              dcl i address;
84 2              f$i$indices$base = last$f$i$adr + size(file$info);
85 2              if filesfound < 2 then
86 2                  return;
87 2              if shr((x$i$adr - f$i$indices$base), 1) < filesfound then
88 2                  do;
89 3                      call print(, ('Not Enough Memory for Sort', cr, lf, '$'));
90 3                      return;
91 3                  end;
92 2              do i = 0 to filesfound - 1;
93 3                  f$i$indices(i) = i; /* initialize f$i$indices */
94 3              end;
95 2              call print(, (cr, lf, 'Sorting Directory...', cr, lf, '$'));
96 2              call qsort(0, filesfound - 1);
97 2              sorted = true;
98 2          end sort;

99 1          end sort;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0366H      870D
VARIABLE AREA SIZE = 0053H      83D
MAXIMUM STACK SIZE = 0006H      6D
150 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TIMESTAMP

OBJECT MODULE PLACED IN TIMEST

COMPILER INVOKED BY: PLM80 TIMEST.PLH DEBUG PAGEWIDTH(130) OPTIMIZE OBJECT(TIMEST)

```

1      $title('SDIR - Display Time Stamps')
      timestamp;
      do;
          /* Display time stamp module for extended directory */
          /* Time & Date ASCII Conversion Code          */
          /* From MP/M 1.1 TOD program                    */

          $include(comlit.lit)

          =
2      1 = declare
          =      lit      literally      'literally',
          =      dcl      lit      'declare',
          =      true     lit      'Offh',
          =      false    lit      '0',
          =      boolean  lit      'byte',
          =      forever  lit      'while true',
          =      cr       lit      '13',
          =      lf       lit      '10',
          =      tab      lit      '9',
          =      ctrlc    lit      '3',
          =      ff       lit      '12',
          =      page$len$offset lit      'lch',
          =      nopage$mode$offset lit      '2Ch',
          =      sectorlen lit      '128';

3      1      print$char: procedure (char) external;
4      2          declare char byte;
5      2      end print$char;

6      1      terminate: procedure external;
7      2      end terminate;

8      1      declare tod$adr address;
9      1      declare tod based tod$adr structure (
          opcode byte,
          date address,
          hrs byte,
          min byte,
          sec byte,
          ASCII (21) byte );

10     1      declare string$adr address;
11     1      declare string based string$adr (1) byte;
12     1      declare index byte;

13     1      emitchar: procedure(c);
14     2          declare c byte;
15     2          string(index := index + 1) = c;
16     2      end emitchar;

17     1      emitn: procedure(a);

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

18 2      declare a address;
19 2      declare c based a byte;
20 2      do while c < '$';
21 3          string(index := index + 1) = c;
22 3          a = a + 1;
23 3      end;
24 2      end emitn;

25 1      emit$bcd: procedure(b);
26 2          declare b byte;
27 2          call emitchar('0'+b);
28 2      end emit$bcd;

29 1      emit$bcd$pair: procedure(b);
30 2          declare b byte;
31 2          call emit$bcd(shr(b,4));
32 2          call emit$bcd(b and 0fh);
33 2      end emit$bcd$pair;

34 1      emit$colon: procedure(b);
35 2          declare b byte;
36 2          call emit$bcd$pair(b);
37 2          call emitchar(':');
38 2      end emit$colon;

39 1      emit$bin$pair: procedure(b);
40 2          declare b byte;
41 2          call emit$bcd(b/10); /* makes garbage if not < 10 */
42 2          call emit$bcd(b mod 10);
43 2      end emit$bin$pair;

44 1      emit$slant: procedure(b);
45 2          declare b byte;
46 2          call emit$bin$pair(b);
47 2          call emitchar('/');
48 2      end emit$slant;

49 1      declare
          base$year lit '78', /* base year for computations */
          base$day lit '0', /* starting day for base$year 0..6 */
          month$days (*) address data
          /* jan feb mar apr may jun jul aug sep oct nov dec */
          ( 000,031,059,090,120,151,181,212,243,273,304,334);

50 1      leap$days: procedure(y,m) byte;
51 2          declare (y,m) byte;
          /* compute days accumulated by leap years */
52 2          declare yp byte;
53 2          yp = shr(y,2); /* yp = y/4 */
54 2          if (y and 11b) = 0 and month$days(m) < 59 then
          /* y not 00, y mod 4 = 0, before march, so not leap yr */
55 2              return yp - 1;
          /* otherwise, yp is the number of accumulated leap days */
56 2          return yp;
57 2      end leap$days;

58 1      declare word$value address;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

59 1  get$next$digit: procedure byte;
      /* get next lsd from word$value */
60 2  declare lsd byte;
61 2  lsd = word$value mod 10;
62 2  word$value = word$value / 10;
63 2  return lsd;
64 2  end get$next$digit;

65 1  bcd:
      procedure (val) byte;
66 2  declare val byte;
67 2  return shl((val/10),4) + val mod 10;
68 2  end bcd;

69 1  declare (month, day, year, hrs, min, sec) byte;

70 1  bcd$pair: procedure(a,b) byte;
71 2  declare (a,b) byte;
72 2  return shl(a,4) or b;
73 2  end bcd$pair;

74 1  compute$year: procedure;
      /* compute year from number of days in word$value */
75 2  declare year$length address;
76 2  year = base$year;
77 2  do while true;
78 3  year$length = 365;
79 3  if (year and 11b) = 0 then /* leap year */
80 3  year$length = 366;
81 3  if word$value <= year$length then
82 3  return;
83 3  word$value = word$value - year$length;
84 3  year = year + 1;
85 3  end;
86 2  end compute$year;

87 1  declare
      week$day byte, /* day of week 0 ... 6 */
      day$list (*) byte data
      ('Sun$Mon$Tue$Wed$Thu$Fri$Sat$'),
      leap$bias byte; /* bias for feb 29 */

88 1  compute$month: procedure;
89 2  month = 12;
90 2  do while month > 0;
91 3  if (month := month - 1) < 2 then /* jan or feb */
92 3  leap$bias = 0;
93 3  if month$days(month) + leap$bias < word$value then return;
95 3  end;
96 2  end compute$month;

97 1  declare
      date$test byte, /* true if testing date */
      test$value address; /* sequential date value under test */

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SFR. # _____

```

98  1      get$date$time; procedure;
        /* get date and time */
99  2      hrs = tod.hrs;
100 2      min = tod.min;
101 2      sec = tod.sec;
102 2      word$value = tod.date;
        /* word$value contains total number of days */
103 2      week$day = (word$value + base$day - 1) mod 7;
104 2      call compute$year;
        /* year has been set, word$value is remainder */
105 2      leap$bias = 0;
106 2      if (year and 11b) = 0 and word$value > 59 then
107 2          /* after feb 29 on leap year */ leap$bias = 1;
108 2      call compute$month;
109 2      day = word$value - (month$days(month) + leap$bias);
110 2      month = month + 1;
111 2      end get$date$time;

112 1      emit$date$time; procedure;
113 2      if tod.opcode = 0 then
114 2          do;
115 3          call emitn(.day$list(shl(week$day,2)));
116 3          call emitchar(' ');
117 3          end;
118 2      call emit$slant(month);
119 2      call emit$slant(day);
120 2      call emit$bin$pair(year);
121 2      call emitchar(' ');
122 2      call emit$colon(hrs);
123 2      call emit$colon(min);
124 2      if tod.opcode = 0 then
125 2          call emit$bcd$pair(sec);
126 2      end emit$date$time;

127 1      tod$ASCII:
        procedure (parameter);
128 2      declare parameter address;
129 2      declare ret address;

130 2      ret = 0;
131 2      tod$adr = parameter;
132 2      string$adr = .tod.ASCII;
133 2      if (tod.opcode = 0) or (tod.opcode = 3) then
134 2          do;
135 3          call get$date$time;
136 3          index = -1;
137 3          call emit$date$time;
138 3          end;
        else
139 2          call terminate;          /* error */
140 2      end tod$ASCII;

141 1      declare lcltod structure (
        opcode byte,
        date address,
        hrs byte,
        min byte,

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

PL/M-80 COMPILER SDIR - DISPLAY TIME STAMPS

```

        sec byte,
        ASCII (21) byte );

142  1    display$time$stamp: procedure (tsadr) public;
143  2        dcl tsadr address,
            i byte;

144  2        lcltod.opcode = 3;    /* display time and date stamp, no seconds */
145  2        call move (4,tsadr,.lcltod.date);    /* don't copy seconds */

146  2        call tod$ASCII (.lcltod);
147  2        do i = 0 to 13;
148  3            call printchar (lcltod.ASCII(i));
149  3        end;
150  2    end display$time$stamp;

151  1    dcl last$data$byte byte initial(0);

152  1    end timestamp;

```

MODULE INFORMATION:

```

CODE AREA SIZE    = 035CH    860D
VARIABLE AREA SIZE = 0046H    70D
MAXIMUM STACK SIZE = 000CH    12D
241 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

COPYRIGHT © 1982 DIGITAL RESEARCH P. O. BOX 579 PACIFIC GROVE, CA 93950 SER. # _____

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE UTILITY

OBJECT MODULE PLACED IN UTIL

COMPILER INVOKED BY: PLM80 UTIL.PLM DEBUG PAGewidth(130) OPTIMIZE OBJECT(UTIL)

```

        $title('SDIR - Utility Routines')
1      utility:
        lo;

        /* Utility Module for SDIR */

        $include(comlit.lit)

        =
2      1 = declare
        =      lit            literally      'literally',
        =      dcl            lit            'declare',
        =      true           lit            '0ffh',
        =      false          lit            '0',
        =      boolean         lit            'byte',
        =      forever        lit            'while true',
        =      cr              lit            '13',
        =      lf              lit            '10',
        =      tab             lit            '9',
        =      ctrlc           lit            '3',
        =      ff              lit            '12',
        =      page$len$offset lit            '1ch',
        =      nopage$mode$offset lit        '2Ch',
        =      sectorlen       lit            '128';

        /* ----- arithmetic functions ----- */

3      1      add3byte: procedure(byte3adr,num) public;
4      2          dcl (byte3adr,num) address,
                b3 based byte3adr structure (
                lword address,
                hbyte byte),
                temp address;

5      2          temp = b3.lword;
6      2          if (b3.lword := b3.lword + num) < temp then          /* overflow */
7      2              b3.hbyte = b3.hbyte + 1;
8      2      end add3byte;

                /* add three byte number to 3 byte value structure */
9      1      add3byte3: procedure(totalb,numb) public;
10     2          dcl (totalb,numb) address,
                num based numb structure (
                lword address,
                hbyte byte),
                total based totalb structure (
                lword address,
                hbyte byte);

11     2          call add3byte(totalb,num.lword);
12     2          total.hbyte = num.hbyte + total.hbyte;

```

CP/m plus V3.1

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # *CP3-135*

```

13  2      end add3byte3;

                                     /* divide 3 byte value by 8 */
14  1      shr3byte: procedure(byte3adr) public;
15  2          dcl byte3adr address,
                b3 based byte3adr structure (
                lword address,
                hbyte byte),
                temp1 based byte3adr (2) byte,
                temp2 byte;

16  2          temp2 = ror(b3.hbyte,3) and 11100000b; /* get 3 bits */
17  2          b3.hbyte = shr(b3.hbyte,3);
18  2          b3.lword = shr(b3.lword,3);
19  2          temp1(1) = temp1(1) or temp2;          /* or in 3 bits from hbyte */
20  2      end shr3byte;

```

```

                                     /* ----- print routines ----- */
21  1      moni: procedure(f,a) external;
22  2          declare f byte, a address;
23  2      end moni;

24  1      break: procedure external;
25  2      end break;

```

```

#include(fcb.lit)

```

```

26  1  = declare
      =      f$drvusr      lit '0',          /* drive/user byte */
      =      f$name        lit '1',          /* file name */
      =      f$namelen     lit '8',          /* file name length */
      =      f$type        lit '9',          /* file type field */
      =      f$typelen     lit '3',          /* type length */
      =      f$rw          lit '9',          /* high bit is R/W attribute */
      =      f$dirsyz      lit '10',         /* high bit is dir/sys attribute */
      =      f$arcc        lit '11',         /* high bit is archive attribute */
      =      f$ex          lit '12',         /* extent */
      =      f$sl          lit '13',         /* module byte */
      =      f$rc          lit '15',         /* record count */
      =      f$diskmap     lit '16',         /* file disk map */
      =      diskmaplen    lit '16',         /* disk map length */
      =      f$drvusr2     lit '16',         /* fcb2 */
      =      f$name2       lit '17',
      =      f$type2       lit '25',
      =      f$rrcc        lit '33',         /* random record */
      =      f$rrcco       lit '35',         /* ' ' overflow */
      =

```

```

/* BIOS calls */

```

```

27  1      print$char: procedure(char) public;
28  2          declare char byte;
29  2          call moni(2,char);
30  2      end print$char;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

31 1  print; procedure(string$adr) public;
32 2      dcl string$adr address;
33 2      call mon1(9,string$adr);
34 2  end print;

35 1  printb; procedure public;
36 2      call print$char(' ');
37 2  end printb;

38 1  crlf; procedure public;
39 2      call print$char(cr);
40 2      call print$char(lf);
41 2  end crlf;

42 1  printfn; procedure(fname$adr) public;
43 2      dcl fname$adr address,
          file$name based fname$adr (1) byte,
          i byte;
                                     /* <filename> ' ' <filetype> */

44 2      do i = 0 to f$namelen - 1;
45 3          call printchar(file$name(i) and 7fh);
46 3      end;
47 2      call printchar(' ');
48 2      do i = f$namelen to f$namelen + f$typepelen - 1;
49 3          call printchar(file$name(i) and 7fh);
50 3      end;
51 2  end printfn;

52 1  pdecimal; procedure(v,prec,zerosup) public;
                                     /* print value v, field size = (log10 prec) + 1 */
                                     /* with leading zero suppression if zerosup = true */
53 2      declare v address,           /* value to print */
          prec address,               /* precision */
          zerosup boolean,           /* zero suppression flag */
          d byte;                    /* current decimal digit */

54 2      do while prec <> 0;
55 3          d = v / prec;             /* get next digit */
56 3          v = v mod prec;           /* get remainder back to v */
57 3          prec = prec / 10;         /* ready for next digit */
58 3          if prec <> 0 and zerosup and d = 0 then
59 3              call printb;
          else
60 3              do;
61 4                  zerosup = false;
62 4                  call printchar('0'+d);
63 4              end;
64 3          end;
65 2  end pdecimal;

66 1  p3byte; procedure(byte3adr,prec) public;
                                     /* print 3 byte value with 0 suppression */
67 2      dcl byte3adr address,       /* assume high order bit is < 10 */
          prec address,
          b3 based byte3adr structure (
              lword address,
              hbyte byte),

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

        i byte;

68  2      if b3.hbyte <> 0 then          /* prec = 1 for 6 chars, 2 for 7 */
69  2      do;
70  3          call pdecimal(b3.hbyte,prec,true); /* 3 for 8 chars printed */
71  3          call pdecimal(b3.lword,10000,false);
72  3      end;
73  2      else
74  3      do;
75  3          i = 1;
76  4          do while i <= prec;
77  4              call printb;
78  4              i = i * 10;
79  3          end;
80  3          call pdecimal(b3.lword,10000,true);
81  2      end p3byte;

82  1      end utility;

```

MODULE INFORMATION:

CODE AREA SIZE	= 01FFH	511D
VARIABLE AREA SIZE	= 001EH	30D
MAXIMUM STACK SIZE	= 0008H	8D
185 LINES READ		
0 PROGRAM ERROR(S)		

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

0000 DIR					
0000 MCD80A					
0004 BDISK	0060 BUFF	0080 BUFFA	0050 CMDRV	0000 CPU	
007C CR	006D DOLLA	005C FCB	006C FCB16	005C FCBA	
005C IFCB	005C IFCBA	0003 IOBYTE	0053 LENO	0056 LEN1	
0006 MAXB	0006 MEMSIZ	0005 MON1	0005 MON2	0005 MON2A	
0005 MON3	0000 OFFSET	006E PARMA	0051 PASS0	0054 PASS1	
007F RO	007D RR	007D RRECA	007F RRECO	005C SFCB	
0080 TBUFF					
0000 SDIR					
39FC MEMORY	024F PLM	36F5 PATCH	37F5 DEBUG	37F6 OS	
37F7 BDOS	37F8 FIND	3800 NUMSEARCHFILES		3801 NOPAGEMODE	
3802 SEARCH	3884 GETALLDIRENTRIES		3885 FIRSTPASS	3886 USRVECTOR	
3888 ACTIVEUSRVECTOR		388A DRVVECTOR	388C FORMAT	388D PAGELEN	
388F MESSAGE	3890 FORMFEEDS	3891 DATEOPT	3892 DISPLAYATTRIBUTES		
3893 SORTOP	3894 CURUSR	3895 CURDRV	041D GETVERSION		
0429 SELECTDRIVE		3896 D	0439 GETCURDRV	0442 GETLOGIN	
0448 GETUSR	0454 GETSCBBYTE	3897 OFFSET	3898 SCBPB		
046C SETCONSOLEMODE		0475 TERMINATE	047E NUMBER	389C CHAR	
0496 MAKENUMERIC		389D CHARADR	389F LEN	38A0 VALADR	
38A2 PLACE	38A4 I	0515 SETVEC	38A5 VADR	38A7 NUM	
054D BITLOC	38A8 VECTOR	38AA I	058E GETNXT	38AB VECTORADR	
38AD I	38AE MASK	38B0 PCB	38BA GOTOPTIONS	05DB GETOPTIONS	
38BB TEMP	0A3F OPERR	0A49 GETFILESPEC		38BC I	
0B39 SETDEFAULTS		38BD SAVEUVEC	38BF TEMP	38C1 I	
38C2 LASTDSEGBYTE					
0000 SCANNER					
39FC MEMORY	38C3 DEBUG	38C4 D	0D6E WHITESPACE	38C5 STRADR	
38C7 I	0DA2 DELIMITER	38C8 CHAR	0E00 DEBLANK	38C9 BUFADR	
38CB DEST	38CD I	38CE NUMSPACES	38CF STRING	1024 UPPERCASE	
38D0 BUFADR	38D2 I	38D3 PCBBASE	38D5 SCANADR	38D7 INPTR	
38D8 TOKENADR	38DA TPTR	38DB CHAR	38DC NXTCHAR	38DD TCOUNT	
1088 DIGIT	38DE CHAR	10A0 LETTER	38DF CHAR	1088 EATCHAR	
10DD PUTCHAR	38E0 CHARX	1105 GETIDENTIFIER		38E1 MAX	
11CA FILECHAR	38E2 X	1211 EXPANDWILDCARDS		38E3 FIELD SIZE	
38E4 I	38E5 LEFTOVER	38E6 SAVEINPTR	12B5 GETFILESPEC		
38E8 I	13A6 GETNUMERIC	38E9 MAX	1459 GETSTRING	38EA MAX	
14E5 GETTOKENALL		38EB SAVEINPTR	152D GETMODIFIER		
1579 GETOPTION	15B3 GETPARAM	38EC GOTATOKEN	38ED PARENS	15FF ENDSTATE	
1629 STATE8	16B4 STATE7	170A STATE6	174D STATES	179F STATE4	
1843 STATE3	189B STATE2	192F STATE1	199C STARTSTATE	1A08 SCAN	
38EE PCBADR	38F0 STATUS	1B1D SCANINIT	38F1 PCBADR		
0000 SEARCH					
39FC MEMORY	38F3 USEDDE	38F5 FILESFOUND	38F7 DIRLABEL	1B90 READCHAR	
1B99 CHECKCONSOLESTATUS		1BA2 SEARCHFIRST		38F8 FCBADDRESS	
1BB2 SEARCHNEXT	1BBB BREAK	38FA X	38FB BUFFCBADR	38FD FIRSTFIADR	
38FF FIADR	3901 LASTFIADR	3903 SFCBADR	3905 SFCRSPRESENT		
3906 XIADR	1BCC COMPARE	3908 LENGTH	3909 STR1ADR	390B STR2ADR	
390D I	1C27 MATCH	390E I	390F TEMP	39FC HASHTABLE	
3911 HASHENTRYADR		1D1E HASHLOOKUP	3913 I	3914 FOUND	
3915 HASHINDEX	1DE7 STOREFILEINFO		3916 I	3917 J	
3918 DMAPCNT	3919 TEMP	1F37 STOREFILE	208E GETFILES	391B DCNT	
2189 SEARCHINIT					
0000 SORT					
39FC MEMORY	391C SORTED	391D MIDADR	21F5 MULT23	391F INDEX	
221E LESSTHAN	3921 STR1ADR	3923 STR2ADR	3925 I	3926 C1	
3927 C2		392B FIINDICESBASE		2270 QSORT	392A L
392C R	392E I	3930 J	3932 TEMP	3934 STACK	
396C SP		248C SORT	396D I		
0000 DISPLAY					
39FC MEMORY	396F FILEDISPLAYED		3970 CURFILE	3972 FCBADR	
2700 DIRECTCONSOLEIO		3974 FIRSTTIME	2709 WAITKEYPRESS		
3974 CHAR	3977 GLOBAL INFCOUNT		272C CRLEANDCHECK		

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

3978 TOTALBYTES	3978 TOTALRECS	397E TOTALKBLOCKS
275F ADDTOTALS 3981 FILESPERLINE	3982 CURLINE	250B HDR
2533 HDRBARS 255B HDRPU	2572 HDRXFCBBARS	2599 HDRACCESS
25AA HDRCREATE 278E DISPLAYFILEINFO	289F DISPLAYXFCBINFO	
3984 FIRSTTITLE	296D DISPLAYTITLE	29C5 SHORTDISPLAY
3985 FNAMEADR 2A4A TESTATT	3987 CHAR	3988 OFF 3989 ON
2A82 RIGHTATTRIBUTES	398A NAMEADR	2AB5 SHORTDIR 398C DCNT
398D LASTPLUSONE	398F INDEX	2B36 GETNXTFILEINFO
3991 RIGHTUSR 2BE3 SIZEDISPLAY	2C6A DISPLAYNODIRLABEL	
2D72 DISPLAYWITHDIRLABEL	2E65 DISPLAYFILES	
0000 UTILITY		
39FC MEMORY 303C ADD3BYTE	3992 BYTE3ADR	3994 NUM 3996 TEMP
306F ADD3BYTE3 3998 TOTALB	399A NUMB	309B SHR3BYTE 399C BYTE3ADR
399E TEMP2 30D2 PRINTCHAR	399F CHAR	30E2 PRINT 39A0 STRINGADR
30F2 PRINTB 30F8 CRLF	3100 PRINTFN	39A2 FNAMEADR 39A4 I
315B PDECIMAL 39A5 V	39A7 PREC	39A9 ZEROSUP 39AA D
31CD P3BYTE 39AB BYTE3ADR	39AD PREC	39AF I
0000 DPB80		
39FC MEMORY 39B0 KPERBLOCK	39B2 DPBBASE	323B DPBBYTE 39B4 PARAM
324B DPBWORD 39B5 PARAM	3279 BASEDPB	
0000 TIMESTAMP		
39FC MEMORY 39B6 TODADR	39B8 STRINGADR	39BA INDEX 32CA EMITCHAR
39BB C 32E1 EMITN	39BC A	3310 EMITBCD 39BE B
331E EMITBCDPAIR	39BF B	3339 EMITCOLON 39C0 B
334A EMITBINPAIR	39C1 B	336F EMITSLANT 39C2 B
3296 MONTHDAYS 3380 LEAPDAYS	39C3 Y	39C4 M 39C5 YP
39C6 WORDVALUE 33BC GETNEXTDIGIT		39C8 LSD 33DA BCD
39C9 VAL 39CA MONTH	39CB DAY	39CC YEAR 39CD HRS
39CE MIN 39CF SEC	33FD BCDPAIR	39D0 A 39D1 B
340F COMPUTEYEAR	39D2 YEARLENGTH	39D4 WEEKDAY 32AE DAYLIST
39D5 LEAPBIAS 344D COMPUTEMONTH		39D6 DATETEST 39D7 TESTVALUE
348B GETDATETIME	3509 EMITDATETIME	3561 TODASCII
39D9 PARAMETER 39DB RET	39DD LCLTOD	35A3 DISPLAYTIMESTAMP
39FB TSADR 39FA I	39FB LASTDATABYTE	

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

0000 DIR#

0000 MCD80A#

0000 SDIR#

041D 64	041D 65	0429 66	0429 67	042D 69
0438 70	0439 76	0439 77	0442 78	0442 79
0442 80	044B 81	044B 82	044B 83	0454 84
0454 85	0458 88	045E 89	0463 90	046C 91
046C 92	046C 93	0474 94	0475 95	0475 96
047D 97	047E 98	0482 100	0496 101	0496 102
04A5 104	04AE 105	04B4 106	04C2 107	04DB 108
04DB 109	0502 110	050D 111	0512 112	0515 113
0515 114	051D 116	0525 117	0535 118	054C 119
054D 120	0553 122	0558 123	0578 124	0585 125
0587 126	058A 127	058E 128	058E 129	0594 131
05A7 132	05AA 133	05B0 134	05B9 135	05C8 136
05D7 137	05DB 138	05DB 144	05DB 146	05FA 147
0602 149	060C 150	0614 151	0631 152	0639 153
0656 155	065B 156	0660 157	0663 158	066D 159
0675 160	067F 162	068A 163	0692 164	069D 165
06A5 166	06A8 167	06AB 168	06B5 170	06BD 171
06CB 172	06EE 173	0705 174	0712 175	0715 176
0718 177	0722 178	072A 179	0734 181	0741 182
0749 183	0756 184	075E 185	0768 186	0773 187
0776 188	0779 189	0796 190	079E 191	07BB 192
07C3 193	07CD 195	07D8 196	07E0 197	07EB 198
07F3 199	07FE 200	0806 201	0809 202	080C 203
0816 204	081E 205	0821 206	0827 207	082A 209
0834 211	083A 212	0844 213	0857 214	0862 215
0865 216	086E 217	0874 218	0877 219	087A 220
0884 222	088A 223	08A2 224	08A5 225	08C4 226
08E1 227	08EA 228	0902 230	090A 231	0917 232
0937 233	094E 234	095B 235	095E 236	0961 237
0964 238	096A 239	096D 240	0970 241	0997 243
099D 244	09A7 245	09AA 246	09C9 247	09E6 249
09EC 250	09FA 251	09FD 252	0A13 253	0A24 254
0A27 255	0A2D 256	0A30 257	0A33 258	0A36 259
0A36 260	0A39 261	0A3E 262	0A3F 263	0A45 264
0A48 265	0A49 266	0A49 268	0A51 270	0A77 271
0AB1 272	0AC9 273	0ADE 274	0AE7 275	0AFB 276
0B13 277	0B1A 278	0B1D 280	0B23 281	0B2F 282
0B32 283	0B32 284	0B38 285	0B39 286	0B39 287
0B45 288	0B4D 289	0B59 290	0B61 291	0B6C 293
0B74 294	0B79 295	0B7C 296	0B84 297	0B8C 299
0BA1 300	0BB2 301	0BB7 302	0BB7 303	0BC3 304
0BD3 305	0BE8 306	0BFB 307	0C11 308	0C1E 309
0C2E 310	0C61 312	0C67 313	0C6A 314	0C6A 315
0C74 316	0C80 317	0C8A 318	0C93 320	0C9B 322
0CA8 324	0CB3 325	0CBE 326	0CC4 327	0CC4 328
0CC4 329	0CC4 330	024C 334	0252 335	0259 336
0260 337	0276 339	027C 340	027F 341	0282 342
0285 343	028B 344	0291 345	0297 346	029F 347
02A4 348	02B1 349	02BB 350	02C3 351	02C9 353
02CF 354	02D2 355	02D5 356	02DF 357	02E5 359
02EB 360	02EE 361	02EE 362	02F1 363	02F4 364
02F7 365	0305 366	030C 367	0312 368	0318 369
0321 370	0326 371	0342 373	0347 374	0350 375
0356 376	035E 377	0365 378	0368 379	036E 380
0387 381	038C 382	038C 383	0391 384	0397 385
039F 386	03A9 387	03BB 389	03D3 391	03D6 392
03DB 393	03E0 394	03E7 395	03EA 396	03EA 397
03ED 398	03ED 399	03F6 400	03F9 401	03FF 402
0402 403	0412 404	0418 405	041B 407	

0000 SCANNER#

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

0D6E 26				
0D74 28	0D79 29	0D97 30	0D9B 31	0D9E 32
0DA2 33	0DA2 34	0DA6 36	0DFA 37	0DFD 38
0E00 39	0E00 41	0E06 43	0E0B 44	0E1E 45
0E47 46	0E4C 47	0E5C 48	0E93 49	0EA3 51
0EAB 52	0EB4 53	0EB4 54	0EBB 55	0EBE 56
0ED6 57	0EE6 58	0EFA 59	0F2C 61	0F38 62
0F3D 63	0F3D 64	0F44 65	0F47 66	0F60 68
0F6C 69	0F7D 70	0F86 71	0FAF 72	0FB6 73
0FE6 74	0FF6 75	0FFB 76	1002 77	1002 78
1005 79	1016 80	1023 81	1024 82	102A 84
102F 85	103F 86	1065 87	107D 88	1084 89
1087 90	1088 99	108C 101	10A0 102	10A0 103
10A4 105	10B8 106	10B8 107	10B8 108	10CA 109
10DC 110	10DD 111	10E1 113	10F2 114	1104 115
1105 116	1109 118	110E 119	1126 120	112A 121
1165 122	116C 123	116F 124	1176 125	1179 126
11A7 127	11AA 128	11B1 129	11B4 130	11BD 131
11C6 132	11CA 133	11CA 134	11CE 136	1211 137
1211 138	1215 140	121D 141	1234 142	123C 144
1242 145	124A 146	124D 147	1258 148	125F 149
1262 150	1265 151	126F 153	1277 154	127A 155
127A 156	128B 157	1290 158	129A 159	12A2 160
12A5 161	12AC 162	12AF 163	12B2 164	12B5 165
12B5 166	12B5 168	12C3 169	12CF 170	12D9 171
12E1 172	12F8 174	1302 175	1305 176	1308 177
130B 178	1311 179	1316 180	1352 181	135B 182
135E 183	1368 184	136B 185	1373 187	1376 188
1381 190	1386 191	1390 192	1393 193	1393 194
1393 195	139C 196	13A3 197	13A6 198	13A6 199
13AA 201	13B6 202	13B9 203	13E1 204	13E8 205
13EB 206	13F9 207	13FC 208	1420 209	142F 211
1436 212	1439 213	1447 214	144A 215	144D 216
1456 217	1459 218	1459 219	145D 221	1465 222
1468 223	146B 224	1492 225	1499 226	149C 227
14AA 228	14AD 229	14C5 230	14C8 231	14CB 232
14D3 233	14D6 234	14DF 235	14E2 236	14E5 237
14E5 238	14E5 240	14EB 241	14F2 242	14F5 243
14FC 244	14FF 245	1504 246	1509 247	1513 248
151D 249	1527 250	152A 251	152D 252	152D 253
152D 254	1551 256	155A 257	155D 258	155D 259
1564 261	1573 262	1576 263	1576 264	1579 265
1579 266	1579 267	157E 268	158A 270	1599 271
15A4 272	15AD 273	15B0 274	15B0 275	15B3 276
15B3 277	15B3 278	15D7 280	15E0 281	15E3 282
15E3 283	15EA 285	15F9 286	15FC 287	15FC 288
15FF 289	15FF 292	15FF 293	1606 295	160F 296
1612 297	1612 298	161B 299	1626 300	1629 301
1629 302	1629 303	1630 305	1638 306	163F 307
163F 308	1647 309	164B 310	1653 312	1656 313
167A 314	167E 315	1686 316	168A 317	168E 318
1691 319	16A9 321	16AC 322	16B0 323	16B0 324
16B4 325	16B4 326	16B4 327	16BB 329	16C3 330
16CA 331	16CA 332	16D2 333	16D6 334	16EE 336
16F1 337	16F5 338	16F8 339	1700 341	1703 342
1707 343	1707 344	170A 345	170A 346	170A 347
1711 349	1719 350	1720 351	1720 352	1727 354
1730 355	1739 356	173C 357	173C 358	1746 359
174A 360	174D 361	174D 362	174D 363	1754 365
175C 366	1763 367	1763 368	176B 370	176E 371
1772 372	1772 373	1779 375	1782 376	178B 377
178E 378	178E 379	1798 380	179C 381	179F 382
179F 383	17A0 385	17A7 387	17AF 388	17B6 389
17B6 390	17BE 391	17C3 392	17CB 393	17CE 394
17E6 395	17EB 396	17F5 397	1819 398	181E 399
1826 400	182B 401	1830 402	183A 403	183F 404

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

1843 405	1843 406	1843 407	184A 409	1852 410
1859 411	1859 412	1860 414	1869 415	1872 416
1875 417	1875 418	1887 419	188A 420	1894 421
1898 422	189B 423	189B 424	189B 425	18A2 427
18AA 428	18B1 429	18B1 430	18C9 431	18D1 432
18D5 433	18DB 434	18DF 435	18E2 436	18EA 438
18ED 439	18F1 440	18F1 441	1915 443	191D 444
1924 445	1927 446	192B 447	192B 448	192F 449
192F 450	192F 451	1936 453	193E 454	1945 455
1945 456	194C 458	1955 459	195E 460	1961 461
1961 462	1969 463	1970 464	1973 465	1976 466
1988 467	198B 468	1995 469	1999 470	199C 471
199C 472	199C 473	19A4 475	19A9 476	19AC 477
19B4 478	19BB 479	19BB 480	19C3 481	19C7 482
19CF 483	19D2 484	19DA 486	19E1 487	19E4 488
19EB 489	19EB 490	19F0 492	19F3 493	19F7 494
19F7 495	1A01 496	1A05 497	1A08 498	1A08 499
1A0E 501	1A14 502	1A20 503	1A2E 504	1A3B 505
1A3B 506	1A40 507	1A49 508	1A52 509	1A5D 510
1A5E 511	1A6B 512	1A74 513	1A81 514	1A8A 515
1A97 516	1AA0 517	1AAD 518	1AB6 519	1AC3 520
1ACC 521	1AD9 522	1AE2 523	1AE7 524	1AEF 525
1AF8 526	1B07 527	1B1C 528	1B1D 529	1B23 531
1B29 532	1B34 533	1B48 534	1B51 535	

0000 SEARCH#

1B90 32				
1B90 33	1B99 34	1B99 72	1B99 73	1BA2 74
1BA2 75	1BA8 77	1BB2 78	1BB2 79	1BB2 80
1BBB 81	1BBB 87	1BBB 89	1BC2 91	1BCB 92
1BCB 93	1BCB 94	1BCC 98	1BDB 100	1BEB 101
1C1A 102	1C1D 103	1C24 104	1C27 105	1C27 106
1C27 108	1C37 109	1C3E 110	1C41 112	1C47 113
1C51 114	1C63 115	1C66 116	1C66 117	1C8E 118
1C98 119	1CA8 120	1CCC 121	1CE5 122	1CEA 123
1D0E 124	1D13 125	1D1A 126	1D1E 127	1D1E 129
1D1E 131	1D23 132	1D31 133	1D43 134	1D4A 135
1D52 136	1D55 137	1D60 138	1D6F 139	1D74 140
1D8C 141	1D84 142	1D8C 144	1DC6 145	1DD1 146
1DD1 147	1DD4 148	1DE0 149	1DE3 150	1DE6 151
1DE7 152	1F37 154	1F37 155	1F4E 156	1F62 157
1F67 158	1F6C 159	1F79 160	1F7E 161	1F9D 162
1FAD 163	1FB5 164	1FCF 165	1FDB 166	1FE2 167
1FE5 168	1FEE 170	2024 171	2067 172	208D 173
208D 174	1DE7 175	1DF0 177	1DF8 179	1E0D 180
1E10 181	1E1C 182	1E1F 183	1E2C 184	1E33 185
1E4A 186	1E54 187	1E95 188	1EA1 189	1EA1 190
1EA1 191	1EA8 193	1EB3 195	1EBC 197	1ECE 199
1EE7 200	1EEA 201	1EF8 202	1F0D 203	1F1D 204
1F1D 205	1F20 206	1F20 207	1F20 208	1F23 210
1F2E 212	1F31 214	1F34 215	1F34 216	1F34 217
1F37 218	208E 219	208E 221	2094 222	20A0 223
20AE 224	20BE 225	20C8 226	20CE 227	20D1 228
20DF 229	20E4 230	20ED 231	20F3 232	20FC 233
2104 234	2109 235	2111 236	2125 237	212C 238
213F 239	2148 241	214F 242	2158 243	2166 244
216D 246	2175 248	217B 249	217C 250	217C 251
217C 252	217C 253	217F 254	2185 255	2188 256
2189 257	2189 258	219B 260	21A1 261	21A4 262
21A4 263				

0000 SORT#

21F5 10	21FB 12	221E 13	221E 14	
2228 16	2236 17	225D 18	2266 19	226D 20
2270 21	2270 23	227A 25	227F 26	2285 27
228B 28	2293 29	22A6 30	22B3 31	22B7 32
22C3 33	22C9 34	22CF 35	22EB 36	22F7 37
2309 38	2319 39	2320 40	2332 41	2335 42

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

2347 43	2357 44	235E 45	2370 46	2373 47
237F 49	238F 50	23AA 51	23BC 52	23C3 53
23CE 54	23D5 55	23D5 56	23D8 57	23F2 59
23FE 61	2402 62	2414 63	2429 64	2429 65
242F 66	2432 68	243E 70	2442 71	2454 72
2469 73	2469 74	246F 75	246F 76	2472 77
2475 78	247D 79	2486 80	248B 81	248C 82
248C 84	2496 85	24A1 86	24A2 87	24BA 89
24C0 90	24C1 91	24C1 92	24D5 93	24E7 94
24F4 95	24FA 96	2505 97	250A 98	

0000 DISPLAY#

2700 95

2700 96	2709 97	2709 99	2709 101	270F 102
---------	---------	---------	----------	----------

2717 103	271D 104	2720 105	2728 106	272B 107
272C 109	272C 110	2734 112	2742 114	2748 115
274F 116	2752 117	2757 118	2757 119	2757 120
275A 121	275E 122	275F 124	275F 125	276F 126
277D 127	278D 128	278E 137	278E 138	2797 139
279A 140	27AD 141	27B2 142	27B5 143	27C4 144
27C7 145	27D4 146	27D0 147	27E3 148	27E6 149
27F3 150	27FC 151	2802 152	2805 153	280C 155
2819 156	2822 157	2828 158	282B 160	2838 161
2840 162	2843 163	284D 164	2855 165	2858 166
2863 167	2868 168	286E 169	287B 170	2883 171
2886 172	2893 173	289B 174	289E 175	289E 176
289F 177	289F 178	28AF 180	28B2 181	28C0 182
28CC 183	28D5 184	28E4 185	28ED 186	28FC 187
2905 188	290B 189	290E 190	292E 191	293D 192
2943 193	2946 194	2949 195	2964 196	296C 197
296C 198	296D 200	296D 201	2974 202	297C 203
2983 204	2986 205	298C 206	2995 207	299A 208
29A2 210	29A8 211	29B6 212	29B6 213	29B9 214
29BF 215	29C4 216	29C5 217	29CB 219	29E0 221
2A05 223	2A08 224	2A0B 225	2A0E 226	2A11 227
2A14 228	2A1B 229	2A24 230	2A27 231	2A2A 232
2A30 233	2A38 234	2A3B 235	2A42 236	2A49 237
2A4A 238	2A55 240	2A67 241	2A6A 242	2A7C 243
2A7F 244	2A82 245	2A82 246	2A88 248	2A85 249
2AB5 250	2AB5 252	2ABA 253	2ABF 254	2AC8 255
2AD0 256	2AE4 257	2B0F 258	2B16 259	2B23 260
2B2C 261	2B32 262	2B35 263	2B36 265	2B36 267
2B3B 268	2B42 270	2B49 271	2B5B 272	2B7A 273
2B81 274	2B93 275	2B96 276	2BA3 277	2BA9 278
2BAC 280	2BB6 281	2BD5 282	2BDF 283	2BE2 284
2BE2 285	2BE3 286	2BE3 287	2BED 288	2BF5 289
2BFA 290	2C07 291	2C3E 293	2C41 294	2C4A 295
2C5D 296	2C63 297	2C63 298	2C66 299	2C69 300
2C6A 301	2C6A 302	2C6F 303	2C75 304	2C82 305
2CB9 307	2CCE 309	2CE1 311	2CFD 313	2D00 314
2D03 315	2D06 316	2D0C 317	2D0F 318	2D15 319
2D18 320	2D1E 321	2D21 322	2D27 323	2D2A 324
2D34 325	2D3B 326	2D3E 328	2D41 329	2D48 330
2D48 331	2D4B 333	2D4E 334	2D55 335	2D55 336
2D58 337	2D5B 338	2D5E 339	2D65 340	2D68 341
2D6B 342	2D6B 343	2D6E 344	2D71 345	2D72 346
2D72 347	2D77 348	2D7D 349	2D8A 350	2DC1 352
2DD4 354	2DF0 356	2DF3 357	2DF6 358	2DF9 359
2DFF 360	2E05 361	2E0F 362	2E18 363	2E1E 364
2E21 365	2E27 366	2E2D 367	2E30 368	2E3A 369
2E41 370	2E41 371	2E41 372	2E44 373	2E4B 374
2E4E 375	2E51 376	2E58 377	2E5B 378	2E5E 379
2E5E 380	2E61 381	2E64 382	2E65 383	2E65 384
2E6E 385	2E7E 386	2E87 387	2E93 388	2E9A 389
2EA0 390	2EA3 391	2EAC 393	2EB2 394	2EB5 395
2EB5 396	2EC5 397	2ECB 398	2ED1 399	2ED8 401
2F14 402	2F1A 404	2F20 405	2F23 406	2F23 407

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

2F26 409	2F38 411	2F3B 412	2F3E 414	2F41 415
2F41 416	2F44 417	2F4A 418	2F63 420	2F79 422
2F7E 423	2F83 424	2F86 426	2F89 427	2F8C 428
2F8C 429	2F92 430	2F9B 431	2FA0 432	2FA6 433
2FAF 434	2FB5 435	2FC1 436	2FC7 437	2FD0 438
2FD6 439	2FDF 440	2FE4 441	2FE7 442	2FF3 443
2FF8 444	3007 445	3007 446	3013 448	301A 450
301D 451	3020 452	3026 453	3026 454	3029 455
302C 457	3031 458	303B 459	303B 460	303B 461

0000 UTILITY#

303C 3	3046 5	3050 6	3068 7	306E 8
306F 9	3079 11	3087 12	309A 13	309B 14
30A1 16	30AF 17	30BB 18	30CB 19	30D1 20
30D2 27	30D6 29	30E1 30	30E2 31	30E8 33
30F1 34	30F2 35	30F2 36	30F7 37	30F8 38
30F8 39	30FD 40	3102 41	3103 42	3109 44
3117 45	312B 46	312F 47	3134 48	3142 49
3153 50	315A 51	315B 52	316A 54	3176 55
3184 56	318E 57	319A 58	31B5 59	31BB 61
31C0 62	31C9 63	31C9 64	31CC 65	31CD 66
31D7 68	31E2 70	31F5 71	3204 72	3207 74
320C 75	3218 76	321B 77	3228 78	322B 79
323A 80	323A 81			

0000 DPB80#

323B 18	323F 20	324B 21		
324B 22	324F 24	3279 25	3279 26	3279 27
32B4 28	3295 29			

0000 TIMESTAMP#

32CA 13	32CE 15	32E0 16		
32E1 17	32E7 20	32F0 21	3305 22	330C 23
330F 24	3310 25	3314 27	331D 28	331E 29
3322 31	332F 32	3338 33	3339 34	333D 36
3344 37	3349 38	334A 39	334E 41	335E 42
336E 43	336F 44	3373 46	337A 47	337F 48
3380 50	3386 53	3390 54	33B3 55	33B8 56
33BC 57	33BC 59	33BC 61	33CB 62	33D6 63
33DA 64	33DA 65	33DE 67	33FD 68	33FD 70
3403 72	340F 73	340F 74	340F 76	3414 77
3414 78	341A 79	3424 80	342A 81	3436 82
3437 83	3445 84	3449 85	344C 86	344D 88
344D 89	3452 90	345B 91	3467 92	346C 93
3486 94	3487 95	348A 96	348B 98	348B 99

3496 100	349F 101	34AB 102	34B3 103	34C0 104
34C3 105	34C8 106	34E3 107	34E8 108	34EB 109
3506 110	3508 111	3509 112	3509 113	3512 115
3523 116	3528 117	3528 118	352F 119	3536 120
353D 121	3542 122	3549 123	3550 124	3559 125
3560 126	3561 127	3567 130	356D 131	3573 132
357A 133	3591 135	3594 136	3599 137	359C 138
359F 139	35A2 140	35A3 142	35A9 144	35AE 145
35C2 146	35C8 147	35D6 148	35E7 149	35F1 150

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____